Singapore Management University

Institutional Knowledge at Singapore Management University

Dissertations and Theses Collection (Open Access)

Dissertations and Theses

12-2019

Enhanced gesture sensing using battery-less wearable motion trackers

Huy Vu TRAN
Singapore Management University, hvtran.2014@phdis.smu.edu.sg

Follow this and additional works at: https://ink.library.smu.edu.sg/etd_coll

 Part of the Software Engineering Commons

Citation

TRAN, Huy Vu. Enhanced gesture sensing using battery-less wearable motion trackers. (2019). Dissertations and Theses Collection (Open Access).
Available at: https://ink.library.smu.edu.sg/etd_coll/251

This PhD Dissertation is brought to you for free and open access by the Dissertations and Theses at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Dissertations and Theses Collection (Open Access) by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email library@smu.edu.sg.

ENHANCED GESTURE SENSING USING BATTERY-LESS WEARABLE MOTION TRACKERS

TRAN HUY VU

SINGAPORE MANAGEMENT UNIVERSITY
2019

Enhanced Gesture Sensing using Battery-less Wearable Motion Trackers

by

TRAN Huy Vu

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

Dissertation Committee:

Archan MISRA (Supervisor / Chair)
Professor of Information Systems
Singapore Management University

Rajesh Krishna BALAN
Associate Professor of Information Systems
Singapore Management University

Kotaro HARA
Assistant Professor of Information Systems
Singapore Management University

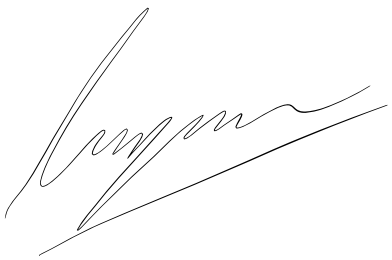
Fahim KAWSAR
Research Director of Pervasive Systems
Nokia Bell-Labs in Cambridge
Design United Professor of IoT
Delft University of Technology

Singapore Management University
2019

Copyright (2019) TRAN Huy Vu

I hereby declare that this PhD dissertation is my original work
and it has been written by me in its entirety.
I have duly acknowledged all the sources of information
which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree
in any university previously.



Tran Huy Vu

11 Dec 2019

Enhanced Gesture Sensing using Battery-less Wearable Motion Trackers

TRAN Huy Vu

Abstract

Wearable devices are gaining in popularity, but are presently used primarily for productivity-related functions (such as calling people or discreetly receiving notifications) or for physiological sensing. However, wearable devices are still not widely used for a wider set of sensing-based applications, even though their potential is enormous. Wearable devices can enable a variety of novel applications. For example, wrist-worn and/or finger-worn devices could be viable controllers for real-time AR/VR games and applications, and can be used for real-time gestural tracking to support rehabilitative patient therapy or training of sports personnel. There are, however, a key set of impediments towards realizing this vision. State-of-the-art gesture recognition algorithms typically recognize gestures, using an explicit initial segmentation step, only after the completion of the gesture, thereby being less appropriate for interactive applications requiring real-time tracking. Moreover, such gesture recognition & hand tracking is relatively energy-hungry and requires wearable devices with sufficient battery capacity. Such battery-driven operation further restricts widespread adoption, as (a) the device must be periodically re-charged, thereby requiring human intervention, and (b) the battery also adds to the wearable device's weight, which potentially affects the wearer's motion dynamics.

In this thesis, I explore the development of new capabilities in wearable sensing along two different dimensions which we believe can help increase the diversity and sophistication of applications and use cases supported by wearable-based systems: (i) Low-latency, low-complexity gesture tracking, and (ii) Ultra-low power or Battery-less operation. The thesis first proposes the development

of a battery-less wearable device that permits tracking of gestural actions by harvesting power from appropriately beamformed WiFi signals. This work requires innovations in both wearable and WiFi AP operations, which work together to support adequate energy harvesting over distances of several meters. Through a combination of simulations and real-world studies, I show that (a) smart WiFi beamforming techniques can help support sufficient energy harvesting by up to 3-4 battery-less devices in a small room, and (b) the prototype battery-less wearable device can support uninterrupted tracking of significant gestural activities by an individual. The thesis then explores the ability of smartwatch to recognize hand gestures early and to track the hand trajectory with low latency, so that it can be used in realizing interactive applications. In particular, I show that our techniques allow a wrist-worn device to be used as a real-time hand tracker and gesture recognizer for an interactive application, such as Table Tennis. The dissertation also demonstrates that my proposed method provides a superior energy-vs-accuracy trade-off compared to more complex gesture tracking algorithms, thereby making it more conducive to operation on battery-less wearable devices. Finally, I evaluate whether my proposed techniques for low-latency gesture recognition can be supported by WiWear-based wearable devices, and establish the set of operating conditions under which such operation is feasible. Collectively, my work advances the state-of-the-art in low-energy wearable-based low-latency gesture recognition, thereby opening up the possible use of battery-less, WiFi-harvesting based devices for gesture-driven applications, especially for sports & rehabilitative training.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would not have been able to complete my Ph.D. journey without the precious support of many individuals.

First of all, I would like to express my deepest gratitude to my advisor, Professor Archan Misra. I have been extremely fortunate to be advised by him, who has patiently listened to me and guided me with his research excellence and enthusiasm. I am grateful to him for his countless support in shaping and refining my research even with the smallest details. I have learned from him not only the research skills but also the working attitude of being committed. I am profoundly indebted to him for always being patient to guide and support me throughout my Ph.D. journey.

I would also like to thank my committee: Associate Professor Rajesh Krishna Balan, Assistant Professor Kotaro Hara and Professor Fahim Kawsar. Their invaluable comments have strengthened my dissertation, and opened my eyes to new opportunities and challenges in my research. I would like to specially thank Fahim who gave me an opportunity to be an intern in Nokia Bell-Labs, a prestigious research lab where I have learned the most advanced technologies.

I would like to thank Assistant Professor Youngki Lee–my co-advisor in the first three years, Richard Davis, Quentin Roy, Kenny Choo for their contributions in my very first pieces of work. My special thanks to Assistant Professor Jie Xiong for his guidance in my recent pieces of work. I would like to thank Chulhong Min for his mentoring and supporting during my internship.

I would like to thank Chew Hong, Pei Huan and Caroline for always being supportive. I thank LiveLabs, MOE fund for financially supporting my research. And thank you to Sipei, Kazae, Shuhui and Jonathan for supporting my experiments and travel.

I have been blessed to have many friends who have added many joyful events to my journey. Thank you to Meera, Kasthuri, Sinh, Loc, Huy, Minh, Sougata, Amit, Kenny and Camelia for always being available for my experiments. I will also like to thank Hirunima, Aritra, Vengat for their help to conduct many experiments.

Finally, I would like to express my special gratitude to my family for their unconditional love and support. Thank you to Ann my beloved wife, who has silently sacrificed her career to be with me through the journey and to give me the best support. I am grateful to my parents for their love, patience and encouragement.

To

Ann my beloved, for always being with me through my journey

&

My parents, for their patience, love and encouragement

Chapter 1

Introduction

In recent years, wearable devices (hereafter referred to as wearables) have gained immense interest from consumers. While smartwatches have been around for a long time (the Seiko TV watch [2] was introduced in 1982), the introduction of embedded sensors has let to a rapid growth in the wearable marketplace, with estimates [4] projecting a market of USD $51.6 Billion by 2022. Wearable is not limited to only wrist-worn devices such as smartwatches, but encompasses a variety of ”smart” devices being worn on any part of a user which can range from smart glasses, smart earbuds to smart clothes, smart tattoo, etc. ”Wearable” now has a specific meaning of wearable computer which must satisfy 3 goals: (1) being mobile, being with the user at all times, (2) being able to augment reality (to assist users), and (3) being able to sense the context [34]. By virtue of its ability to sense the actions of the user, as well as the ambient environments, wearable devices become a powerful platform for digital mediation between an individual and the computing infrastructure.

Wearables enable non-obtrusive sensing applications which were impossible before its emergence. Nowadays, a person can easily use a smartwatch or a smartband to measure his heart rate [1] at any time, which used to require a bulky inflation blood monitor and some skills to operate the equipment. Not limited to only heart rate, many other sensors are now equipped in a wearable

device to measure bio-signals of a user such as blood oxygen saturation (SpO2), blood pressure [17], body temperature [22], electrocardiogram (ECG) [19] etc. Any person can monitor his/her bio-signals in real-time using a wearable to adjust exercise intensity or to track health conditions. Wearable has become an always-available platform which conveniently provides sensing services.

In reality, what makes wearables stand out from mere bio-signal monitors is their motion sensing capability. Inertial sensors, such as accelerometer and gyroscope, have been used for many years in navigation systems in airplanes and ships. Recent technologies have created miniature, low-cost, low-power motion sensors which can be embedded in small form-factor wearables. These sensors, indeed, have revolutionized the way wearable is used. Smartglasses [15, 18, 20], which provide virtual/augmented reality (VR/AR), can determine user head direction to render appropriate video content. Motion sensors in a smartwatch have been used to determine the direction and orientation of user arm, and the smartwatch is used as a pointing device for ubiquitous displays [64, 115], or for VR headsets [69].

Motion sensing also makes wearables smarter, and provides mechanisms for context-aware computing. Motion sensors in wearables reflect user motion-based behaviours such as activities, gestures and interactions. Human activities recognition using mobile and wearable devices has been intensively studied for years. Locomotion modes (such as standing, walking, running) of a user can be inferred from the inertial sensors [102], and used as context information for context-aware applications. Wearable motion sensors have been used to recognize activities of daily living (ADL) such as eating [112], smoking [94], drinking [25] and shopping [111] which are useful for life-logging applications or personalized, context-aware product promotions. Motion sensors also provide information to assess sport/exercise activities such as swimming [28], and to give advice to an athlete/player such as a Virtual Coach [86]. Thank to inertial sensors, wearables can be used to track user hand position (hand tracking)

[113, 83] which can be used as input for VR/AR applications.

Table 1.1: Latency and energy consumption of recent works on motion sensing.

Reference (year)	**Typical sense-making latency (ms)**	**Duration of key gestures /activities (ms)**	**Power consumption footprint (mA)**	**Sensors and sampling frequency (Hz)**
[83](2019)	150	NA (general trajectory)	480	Accelerometer Gyroscope Magnetometer (100)
[113](2016)	10000	NA (general trajectory)	NA	Accelerometer Gyroscope Magnetometer (200)
[36](2016)	NA (Offline)	1000	NA (Offline)	Accelerometer Gyroscope (1000)
[96](2011)	1287	NA	38	Accelerometer Gyroscope (40)
[28](2009)	200	1000 - 5000	29	3 Accelerometers: 1 at wrist, 2 at back (256)

However, to fully utilize the potential of motion sensing in wearables, one needs to overcome several key challenges including: (1) sensor variation, user variation and context variation, (2) energy consumption of inertial sensing on wearable platforms, and (3) latency of the sensor processing pipeline. Table 1.1 summarizes the latency and energy consumption of recent works on motion sensing. The latency mentioned in the table is the latency after the extraction of a segment of data which probably contains the event of interest (e.g, gesture, interaction). It means that the system detects an event of interest 0.15 to 10 seconds after the event occurs. As shown in the table, the power consumption of these methods is exceedingly high compared with recent wearable platforms whose battery capacities can be as low as 40mAh [65]. As summarized in Table 1.1, motion sensing techniques in recent studies have either high latency or high power consumption.

- **Sensor variation, user variation and context variation:** Wearable sensors are limited in size and power consumption to fit in a wearable, so manufacturers have to balance between the accuracy/resolution of the sensors and their size and power consumption based on their own criteria. The technologies different manufacturers use to manufacture sensors are also different. Even the same sensors can behave differently under different situations (e.g, current voltage level, current CPU load), and a CPU can set different sampling rates for the same sensor in different cases. All these factors contribute to variations of sensors which affect accuracy of application results. Moreover, in many motion sensing applications, such as activities recognition, pattern analysis is used to detect patterns of interest. These patterns are known to vary across different subjects (i.e, wearers). Every person has unique physical characteristics such as arm length, body weight, muscle strength, which result in highly different sensor values for the same action. The context in which a user performs an action also greatly changes the amount of noise added into the data. For example, a user performs a *waving* gesture on a bus results in considerably more noise than the same gesture when the user is sitting still. Because of these variations, an activities/gesture recognition model might not achieve sufficiently high accuracy with unseen data.

- **Energy consumption of inertial sensing on wearable platforms:** Wearables such as smartwatches have limited battery capacities to fit their small form-factor. The gradual emergence of even smaller form-factor wearable devices, such as smart ring [70] or smart earbud [65], imposes even stricter limits on battery capacity, and thus, operational lifetime. A study at Washington university in 2015 showed that having to charge a device frequently was one of the reasons people abandon their wearable devices [73]. In addition, the mass production of many small devices makes the battery disposal an alarming threat to environment. The battery needs rare-earth

minerals whose production and eventual disposal are both known to create pollution. Miniature batteries also make the used batteries collection increasingly difficult. Those are the driving reasons that "battery-less" devices are being explored for devices such as small calculator or solar-power smartwatches [78] (which supports recognition of touch gestures). To support motion sensing (a key capability for many gesture-driven applications), a device needs several energy-hungry sensors at the same time such as accelerometer and gyroscope [141, 71, 48] or even including an additional magnetometer [140, 113, 83]. Among those sensors, gyroscope and magnetometer are known to be energy hungry in the context of mobile & wearable sensing [96]. As I shall show later (using numerical analyses), for a low-power wearable device (with off-the-shelf components), to support real-time gesture recognition for a game-playing session lasting one hour, the device needs at least an average continuous power supply of $83\mu W$ throughout a day (24 hours). Any energy harvesting wearable should meet this energy requirement to support fine-grained motion sensing.

- **Latency of the sensor processing pipeline:** While gesture tracking using wearables has gained popularity, our analysis showed that most gesture-based applications currently do not need real-time tracking–e.g., applications such as eating and drinking [25] or swimming [86] analysis currently perform offline analytics; similarly, gesture-based control of home equipment can tolerate 2-5 secs of latency in gesture recognition [16]. Currently developed pipelines for wearable-based inertial sensing data are unsuitable for certain real-time, interactive applications such as fast-paced VR/PC games, sport training applications etc. where feedback is needed even before gesture has finished. Current approaches for gesture recognition need a segmentation step before recognition, which means a system needs to extract a segment of sensor data which probably contains

a gesture, and then classify that segment into a specific gesture or non-gesture. This mechanism implies that the system needs to wait until the end of a gesture, and thus processes a gestural segment only after it finishes. For interactive applications, such a process is too late. Any gesture recognition technique for such real-time interactive applications needs to recognize gestures early ($\sim$20ms after the actual interaction). Of course, to coexist with a future of energy-harvesting based operations, the algorithms should also be (i) computationally simple (to reduce computational energy) and (ii) be robust even with lower sampling rates (because such rates are one way to reduce power consumption).

There are many studies on how to cope with data variation using data augmentation [124, 91], and many studies on how to adapt a model pre-trained on one dataset to another dataset with different distribution (transfer learning and domain adaptation) [68, 47]. However, there is relatively little work on low-latency motion sensing and how to support fine-grained motion sensing using harvested energy on wearables. In this dissertation, I specifically focus on these two challenges, and propose a set of innovations to enable low-latency motion sensing on energy-harvesting wearables.

1.1 Tackling the Challenge: Supporting Low-Latency Gesture Recognition & Tracking on Energy-Harvesting Wearables

Hand tracking using IMU sensors has been studied recently, but those techniques are computationally expensive. For instance, the system in [113] needs to spend 10$\times$T seconds to process a T-second sensor segment on a desktop computer. Though accurate, this technique is inapplicable to real-time tracking problem because of the long processing time, which also results in high power

consumption.

Fortunately, a system does not need to track user's hand all the time to be useful for an interactive application. For example, when a user is playing Table Tennis game, only the hand trajectories during a game gestures are needed to detect whether the user's hand hit a virtual Table Tennis ball or not. During other moments, if the hand is not accurately tracked, it does not affect the game playing ability of the user.

If a system can filter out hand trajectories that does not belong to the supported gestures, the probable space of hand's position will be significantly reduced and easier to be estimated. But firstly, of course, the system must be able to accurately recognize these gestures. Another interesting observation is that different gestures usually have different trajectories. Being able to detect a gesture accurately also provides a hint of the trajectory the hand has gone/will go through.

However, in many interactive applications, even though the system can estimate user's hand trajectory accurately, if the estimation is ready only after the event of interest (e.g, hit a Table Tennis ball), it will severely affect user experience. A technique for early gesture recognition will solve this problem. When combined with gesture-based hand tracking, it enables early gesture recognition & tracking for interactive applications. Early gesture recognition alone can also be applied to "motion correlation" interface [126] to enable low-latency interaction with computer system without pointing or touching.

Low-latency gesture recognition and tracking on an energy-harvesting wearable is only possible if the wearable can harvest sufficient energy from the ambient environment. Though ambient energy sources, in general, are available in many forms, the ambient energy sources for wearable are very limited. One of the very intuitive energy sources is the light. People might think about using photovoltaic cells to harvest energy from light (e.g. sun light or light bulbs). However, in indoor environment, the light energy is very weak. With a cell area

of 4.1cm$\times$2.6cm, the maximum harvested energy is only around $50\mu W$ [21]. The reason is that the light intensity in indoor environment is much lower than the sun light. Also, the photo-voltaic cells rarely catch the light at an incident angle of 90° which gains the highest efficiency. One of the other energy sources which has been exploit for years is motion energy (the Seiko wrist watch). Actually, the average harvested energy from this type of harvester is quite low (an average of $5\mu W$ when worn [93]), and not enough for a wearable with motion sensors. Piezzo electric is another potential energy source, but it is only deployable at places which can cause pressure on the harvester such as in shoes. For wrist-worn or ear-worn devices, the pressure or vibration is not sufficient to generate enough energy to support a motion sensing wearable device.

Energy from Radio Frequency (RF) signal has its own advantages to be used as an energy source for wearables. RF energy has long been used for other purposes such as cooking (e.g. Micro wave oven), and some studies have been done to use RF signal to create a charging cage. But this approach is too complicated for applying it to power always-on wearables. Energy harvesting from ambient radio signal from broadcasting station has also been studied, but it need large antenna to receive sufficient energy for a low-power sensor [97]. Using a completely different philosophy, RF energy is widely used to power RFID tags which are completely mobile and powered on demand. RFID readers have been studied as a possible source for powering small devices such as sensors and low-power cameras [89]. However, dense and ubiquitous deployment of RFID readers is not a very popular option.

Recently, WiFi signal has been studied as a medium to transfer energy to devices, but it is limited to specific deployment of devices and WiFi AP without addressing the problems faced by mobile devices, such as smartphones and wearables. Exploiting WiFi APs to power wearables has several advantages. WiFi APs are ubiquitously deployed in almost every indoor environment, as a means of providing high-speed wireless connectivity. In many urban indoor ar-

eas, WiFi APs/repeaters are densely deployed to provide continuous and high speed services to any device in the area. The WiFi band is commonly used by many low-power devices such as WiFi, Bluetooth, Zigbee etc. Moreover, WiFi AP are widely available commercially.

However, WiFi energy is limited and regulated by laws. Unless it is wisely used, it will not be able to supply enough energy to power wearable devices. For example, a 2.4GHz signal will observe a loss of 35dB at a distance of 1 metre in perfect environment (without multi-path effect). Because people cannot increase the transmission power more than a regulated threshold (30dBm or 1 Watt in the US), the available energy at practically longer distance will be exponentially lower and insufficient for wearable devices. If the APs merely radiate the energy omnidirectionally, the energy is wasted at places where there is no harvesting devices.

Since 2008, with the new WiFi standard 802.11n, MIMO has been officially supported in WiFi APs. MIMO-enabled WiFi AP has multiple antennas to support transmissions of multiple data streams concurrently. Multi-antenna APs enable two distinct key features: (1) being able to estimate angle of arrival (AoA) of the signal, thus the WiFi device, (2) being able to beam-form the transmitted signal to increase the gain in different directions. These two key features can be re-purposed to increase the energy transferred to harvesting devices, by effectively concentrating the transmission power in narrow *beams* directed towards the direction of such wearable and mobile devices.

1.2 Motivating Scenarios

The previous sections show the gap in wearable-based gesture recognition technology, especially in the ability to recognize gestures & track hand motion with ultra-low latency or with low power consumption. Through a survey of various sources, I however discovered a few candidate applications where these capa-

bilities would be critical. Broadly speaking, these applications are interactive in nature, including interactive therapy and VR/AR games:

- In-home Constraint-induced (CI) movement therapy via VR gaming [38] is a promising rehabilitation technology for patients who underwent a trauma or accident which affect their limb function and ability. CI therapy was shown to produce significant improvement in limb use after cerebrovascular accident [120], but a majority of patients were not motivated to use CI therapy because of the restriction in clinical session and devices [92]. Borstad et al. [38] proposed and studied the feasibility of In-home CI therapy using VR game with a Kinect [137] to track users limbs, and encourage patients to use their limb to play games. The study showed that the method was acceptable by patients in the study, and improvements in limb use were observed. Inspired by this rehabilitation method, I depict a rehabilitation scenario where patients do not need to sit/stand in front of a camera, but a wearable device will capture all information of his/her limb movements for assessment.

- The technical innovations, which are used to enable rehabilitation scenario, can also be applied to sport/fitness virtual training applications for new athletes/trainees. A person, who is new to a sport, game or exercise he/she participates in, may find it difficult and unmotivated because of a slow progress or a feeling of being left behind. Eyck et al. studied the effect of Virtual Coach on motivating new athletes [50, 60] to cycle. The result showed that the virtual coach increased the intrinsic and extrinsic motivation. The virtual coach helped participants to bike more in the optimal heart rate zone. Based on these prior studies, I envision a virtual coach application using wearable devices which can assess an athlete/-trainee movements and provide feedback in real-time.

1.2.1 Real-time Monitoring of Interactive Gestures using Battery-less Wearables

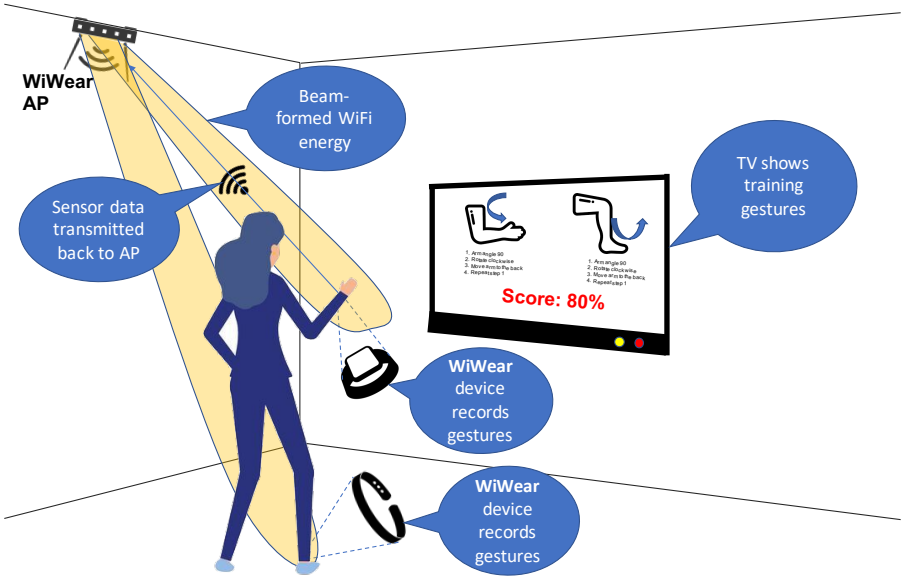


Figure 1.1: A scenario where a patient wears two wearables in a rehabilitation session. The patient looks at a TV screen showing a training gesture, and tries her best to mimic the gesture. The wearables read sensors and transmit the data back to a server through an AP. The server application analyses the data and displays feedback and corrective instructions on the TV screen in real time, allowing the patient to rapidly modify her mistakes. To power the wearables, the AP generates beam-formed WiFi transmissions (RF wave) to focus energy towards the wearables, and the wearables harvest energy from these transmissions.

Use case 1: Rehab Exercises

Thompson clinic is a local health-care center. It provides rehabilitation services for people with different types of traumas. Usually, each patient periodically visits the center and a physician will instruct the patient to perform different activities/exercises. Then a physician observes how the patient carries out her/his movements and gives her/him necessary advice. This is a time consuming and costly process as it requires a physician to be with the patient. Even if the patient wants to practice more (e.g. practice on her/his own at home), there will be no feedback on her/his movements.

The clinics setups an automatic rehabilitation room (illustrated in Figure 1.1)

so that many patients can attend a single session concurrently to save the physician time and effort. Each patient is given a battery-less wearable which contains IMU sensors. The application shows a training video on a TV screen, and the patients try their best to repeat the gesture correctly. The application then collects sensor data from patients in real-time and analyses motion patterns to evaluate if the patient makes correct training gestures. After each gesture, each patient is given instantaneous feedback via several channels such as visual & audio (on the TV screen) and vibration on the wearable. To encourage patients to participate in the exercises, some games are integrated to the training session. Patients can use their gestures to play games such as the one explored in [101].

This application requires low-latency gesture recognition to provide feedback timely to preserve the interactivity of the patients. It also requires hand tracking to assess the similarity or correlation of the actual hand movement and the sample shown on the screen. After each rehabilitation session, the system summarize statistics of each patient, and the physician knows the problems of different patients and give them appropriate advice.

Use case 2: Sport Training

Tom is a teenager. Recently he has been interested in Table Tennis and he wants to be proficient at Table Tennis so that he can join a contest in the city. He usually practices Table Tennis with his coach. But it is costly, so he can spend only 2 hours a week to practice Table Tennis with his coach. He improves his skills slowly, so he wishes to be able to practice more to achieve a certain proficiency, but he could not afford it.

Fortunately, he has a battery-less wristband which he wears every day. He also have a Google Cardboard which he uses to watch 3D movies on his smart phone. He decides to install a Table Tennis training application on his phone. This application, together with his wristband, can provide him free virtual coach service. Whenever he wants to practice Table Tennis, he straps on his Cardboard

(with his phone inside, running the training application). The application first shows him a short training video of a Table Tennis stroke, then it renders a virtual Table Tennis ball flying toward him. Tom then tries to repeat the training video (shown to him previously) to hit the virtual ball. Based on the sensor data from Tom's wristband, the application estimates his hand position, renders a hand avatar in his phone, and determines whether Tom hits the virtual ball or not and gives him audio and vibration feedback instantly. Based on the estimated gesture, the application also assesses his stroke and shows him advice on his phone.

Similar to rehabilitation scenario, this application requires low-latency hand gesture recognition & tracking. More importantly, the feedback must be timely so that he does not feel the lag of the "hit" moment which negatively affects the next stroke immediately after the first one.

Key Challenges

Several challenges need to be tackled to realize the above vision. We utilize the notion of a "virtual table tennis game" as an exemplar of a fast-paced, highly interactive application to elaborate on these challenges.

Recognize the Gesture Before it is Completed: In the case of using game for rehabilitation [101], the real-time feed-backs are very important to improve the experience of patients and make the patients engaged in the session. Moreover, most stroke-based games (e.g., tennis, table tennis and badminton) involve a significant follow-through–i.e., the actual game gesture involves significant movement of the hand both before and after the act of striking of the ball. In fact, as our analysis later (in Section 4) shows, a typical table tennis stroke gesture (the nucleus of gesture) lasts for about 300 msec, with the point of contact between the ball and racket occurring around 120 msec after the start of the stroke (i.e., at approx. the 40%th point of the gesture). Accordingly, we cannot afford to delay the execution of the recognition step till the end of the gesture,

as this would seriously impact the interactive feel of the game.

Calculate the Trajectory Fast and Accurately: For rehabilitation, the trajectory of each gesture is crucial to assess the performance of a patient. Especially, if the rehabilitation process includes gesture-based game playing, the trajectory estimation must be fast enough to support different paces of patients. Similarly, Table Tennis is a very fast-paced game, with professional grade players often exchanging 120 strokes each per minute during a rally [142]. To provide a truly interactive feel, the game must not exhibit lag - i.e., the game must detect the instant of contact between a player's racket and the ball instantly, so that the virtual reality game can proceed apace. This means that we must not only compute the hand's trajectory fast (in real time- e.g., with lag 102 msec corresponding to the 80th percentile of lag tolerance reported later in Figure 4.21 in Section 4), but also precisely (as the calculated point of intersection between the racket and the ball will affect the calculated time instant of contact as well). Such *precise* and *fast tracking* is needed to support lag-free rendering of each user's hand movements during such a game.

1.2.2 Continuous At-Home Gestural Monitoring of Elderly with Battery-less Wearables

Ms. Smith is a retired librarian. She has recently fallen down in her house, and has been hospitalized. Though the hospital discharged her already, the incident has made her unable to walk properly and stably. The physician then applies physiotherapy to improve her ability to walk by asking her to come for a physiotherapy session every week. Each time the physician observes her gait and gives her his advice, and she improves slowly. Both the physician and Ms. Smith wish she had an assisting system to help her practice more at home to help her improve quickly. The physician then askes Ms. Smith to wear an IMU tracking device on her leg, near the ankle. The device continuously records her movement using IMU sensors and sends the data back to a cloud service where

the physician can use an analytical application to analyse the patient gait. Based on the analysis, the physician will call Ms. Smith to give her his advice.

However, because she has trouble with her memory, she often forgets things, especially if the related objects are not on her body. Another problem is that the IMU tracking device is quite bulky and cause discomforts if she wear it throughout a day. One day, she may remove her IMU tracking device for charging and subsequently just forgets to wear it. Alternatively, on another day, she might forget to charge her IMU tracking device, implying that she cannot use it even though it is on her wrist. The worst case is when she does not know the device is out of battery for days. This lack of continuous access to a charged, functional wearable wristband may severely affect her recovering process because the physician does not have daily data to analyse her gait and thus cannot give Ms. Smith proper advice, and thus negatively affect her continuous improvement.

These problems would not manifest themselves if her IMU tracking device harvested energy from the ambient environment. Given such a capability, Ms. Smith would never need to remove her IMU tracking device, and would never have to worry about her device running out of energy. In outdoor environment, the solar energy might be enough for a wearable to operate. However, in indoor environment, ambient energy sources are not always reliable and may not be effective (in this case, the device is covered by her trouser leg). Ideally, a wearable device needs not harvest energy from only one energy source, but should opportunistically harvest from any available source. RF-based energy is a promising source for indoor environment and has been shown to even power low-power cameras [89]. However, unless RFID readers became commonly available in houses, RFID-based charging would not be a practical solution to Ms. Smith's problems. WiFi, on the contrary, is much more ubiquitously available (at least in urban homes), and could be an interesting source of energy for such wearable devices.

Key challenges

Improving Efficiency of WiFi Transmissions for Power Delivery: Through empirical experiments, we observed that the harvested power, from a conventional omnidirectionally transmitting WiFi AP, is too low for practical use (around 1-3μW at distances of 3-4 meters). Note that this problem is particularly acute for WiFi transmissions (which occur in either the 2.4 GHz or 5.5 GHz frequency bands), as opposed to RFID frequencies (865-868 or 902-928 MHz), because it is well known that the wireless propagation loss is greater for higher frequencies [54].

To tackle this problem, we propose to leverage on prior work on (angle-of-arrival) AoA estimation and beamforming to spatially concentrate the transmitted power in the direction of the individual wearable devices. Via experimental studies, we show that we can effectively perform AoA estimation with errors usually less than 5 °and achieve an over 100-fold increase in harvested power.

Supporting Intermittent Operation of Wearable: Our ideal outcome is the design of a wearable device, which utilizes WiFi harvesting to power a relatively high-power inertial sensor used in various gesture-tracking applications. Such a low-power wearable device, worn by a mobile user, gives rise to two challenges: (i) the WiFi AP must be able to track the wearables changing location, without requiring constant active transmissions from the wearable that would consume significant energy, and (ii) the peak power overhead of the wearable system, including the accelerometer and the RF frontend, is over 40 mW—while lower than current commodity watches, this is still much higher than the O(100)micro-W harvested power that RF harvesting is likely to provide. This implies that executing a continuous sensing mode on the wearable is simply infeasible.

To tackle both these challenges, I shall describe an approach, where the wearable usually "sleeps", employing a simple magnetic field tracker to first detect significant motion of the wearable device (such significant motion being

indicative of a possible gestural action). Such significant motion triggers both (i) the transmission of ping packets, which allows the AP to determine the wearables new AoA, and (ii) the activation of the accelerometer sensor, to capture the likely occurrence of meaningful gestures. A supercapacitor helps store the harvested RF energy, and smoothens out transient fluctuations in power supply and drainage. By utilizing experimental studies (with 4 individuals in an office cubicle setting), I show that such a battery-less wearable system can be used to continuously monitor major hand movements, while being net energy-positive.

Support of Multiple Devices/Individuals: For such a wireless charging of individual wearable devices to be practically useful, the infrastructure must also prove to be *scalable*–i.e., it should be able to simultaneously support the operation of multiple wearable devices, either belonging to different individuals or even for situations where a single individual uses multiple wearables. Such a solution would be needed, for example, to cater to multiple occupants in a smart home, or alternately, multiple devices associated with a single individual (e.g., Mrs. Smith may utilize 2 distinct devices, one on each of her legs). The presence of multiple occupants, possibly located in distinct angular directions relative to the AP, gives rise to new tradeoffs in determining beam width and direction-namely, (a) the beamforming process must use narrow beams to increase the power transmitted to each individual device, while (b) a broader beam would, on the other hand, allow multiple devices to be simultaneously covered.

To tackle this problem, I shall propose and develop an optimization framework that can address different objectives–e.g., ensuring max-min fairness or simply enhancing the total power harvested. Via numerical analysis, I provide initial evidence that, by appropriately adapting the spatial and temporal patterns of the RF beams, our AP can support multiple such wearables simultaneously. I also present details on the development and validation of a more elaborate and robust beam adaptation mechanism in my dissertation.

To support the aforementioned scenarios, two key innovations are needed: (1) an energy delivery mechanism to provide sufficient power, unobtrusively, to the wearables so that they can operate the inertial sensors intelligently, and (2) a low-latency low-complexity motion sensing technique which can be deployed on ultra low-power wearables and can produce results in real-time. Based on these considerations, I envision a framework, called *WiWear* , which enables battery-less motion sensing on energy-harvesting wearables.

1.3 *WiWear* Vision

WiWear includes one or more wearables (called *WiWear* device) and one or more AP (called *WiWear* AP). A *WiWear* device harvests energy in the form of WiFi signal from a *WiWear* AP and performs sensing tasks (e.g. gesture, activities recognition).

1.3.1 *WiWear* device

A *WiWear* device supports (but not limited to) motion sensing. As motion sensing requires energy-hungry sensors, if a *WiWear* device can support motion sensing, it is likely to be able to support many other sensing modalities. The *WiWear* device can be worn by a user without having to be removed to recharge. It harvests and accumulates energy continuously throughout the day. Of course, the *WiWear* device can use any other energy harvesting technologies such as light, motion as an opportunistic energy harvesting device. It also uses the stored energy to continuously collect sensor data which will be used for analysis applications such as physical activities level monitoring. At some point, the user can use the *WiWear* device as an interactive input device (e.g. a game controller), which requires real-time streaming of sensor data, to interact with mobile applications (e.g. a rehabilitation application or a VR game).

1.3.2 *WiWear* AP

A *WiWear* AP is a WiFi AP with multiple antennas. There can be multiple *WiWear* APs in an area. The *WiWear* APs fulfill two main goals: (1) fully functioning as a WiFi AP which provides communication medium for normal WiFi devices, and (2) supports energy transfer to *WiWear* devices and sensor data collection from the *WiWear* devices. To smartly transfer energy to *WiWear* devices the *WiWear* APs track the location of *WiWear* devices and boost energy density in the directions of the *WiWear* devices. In the case of a single AP, the angles (or direction) of *WiWear* devices are estimated. In the case of multiple APs, the relative (to the APs) locations of *WiWear* devices are estimated. Based on the locations of *WiWear* devices, the *WiWear* APs modulate and schedule WiFi signal beams to maximize the harvested energy at the *WiWear* devices. Based on the current topology of the devices and the communication demand of normal WiFi devices, the system can also schedule appropriate *WiWear* APs to serve devices efficiently in terms of both energy and communication satisfaction of devices.

1.3.3 Fine-grained motion sensing on battery-less wearables

The *WiWear* APs and *WiWear* devices establish a framework for battery-less motion sensing. A *WiWear* device contains IMU sensors which support both analytic and real-time sensing applications. For analytic applications, the *WiWear* device buffers sensor data locally, and only transfers the data to an *WiWear* AP in batch mode to save energy. For real-time applications, the *WiWear* device transfers each sample to an AP as soon as possible. The *WiWear* device may process the data locally and transfer only the result to an AP depending on the application. For instance, a rehabilitation prepares several battery-less smart bands which can be worn on patient's wrist or leg. Each patient comes to the center, takes one band and wears it. The patient then practices on her/his own by mimicking the sample gestures shown on the TV. After each gesture, the TV shows feed-back of the recent gesture so that the patient can adjust her/his

movements. A physician can access to the statistics of each patient and give her/him appropriate advice.

1.4 Thesis statement

In this thesis, I propose a framework to enable truly battery-less wearables which support fine-grained motion sensing using the energy harvested from WiFi signals. The proposed framework uses a multi-antenna AP to find the angle of the devices and focus the energy radiation toward these angles to boost the harvested energy. I also propose and develop energy-harvesting wearables which harvest energy from WiFi signal and apply "smart" event-based operations to enable energy-hungry motion sensors. Therefore the thesis statement can be stated as follows:

I demonstrate that it is feasible to achieve low-latency motion sensing using battery-less wearables via a combination of:

(a) intelligently beamformed WiFi transmissions, harvested by a battery-less, inertial sensor-embedded, wearable device, such that the harvested power increases sufficiently to permit quasi-intermittent sensor activation;

(b) low-power, inertial sensing based, gesture recognition algorithms that support early gesture recognition and accurate hand trajectory tracking.

Battery-less wearables can enable a variety of applications such as (1) an unobtrusive patient monitoring system using battery-less wearables, and (2) a battery-less real-time gesture recognition & hand tracking for interactive applications. Motivated by the aforementioned vision of battery-less wearable systems, in this thesis, I specifically focus on the following three problems:

1. Develop an energy transfer framework, called *WiWear*, for light-weight battery-less motion sensing wearables. This framework is a foundation for many other novel applications using battery-less wearables. This framework enables smart energy deliver through WiFi signals which is ubiquitously available in most indoor environments and urban spaces. The *WiWear* devices support energy-hungry motion sensors thank to the "smart" event-based operations. The *WiWear* vision does not limit the energy sources to only WiFi, but it rather complements other energy sources so that a truly battery-less devices can be enabled in indoor environments. In Chapter 3, I propose an energy transfer and harvest method using beamformed WiFi signals, and analyse the feasibility of the technique in multiple scenarios. The analyses suggest that the proposed method can support multiple devices (~ 4) concurrently in a range of 3m from the AP. A prototype of the system can transfer up to $400\mu W$ at 1m from the AP and more than $30\mu W$ at 3m from the AP. The experiments on real participants show that the harvested energy is more than the energy expenditure of both system overhead and motion sensing.

2. Develop a real-time gesture recognition and hand tracking algorithm. As discussed previously in this introductory section, motion sensing is one of the most desired applications of wearable devices. Motion sensing is an energy-hungry task, but real-time fine-grained motion sensing is even much more energy hungry. This is one step to explore to what extent the battery-less wearable device can support real-time fine-grained motion sensing. In Chapter 4, I propose a solution to achieve low-latency gesture recognition and hand tracking. The solution includes a HMM model which is specifically designed to avoid segmentation and continuously evaluate the sensor data stream to detect Table Tennis gestures (used as an exemplar of complex, real-time gestures) early, before the gestures end. I also propose a gesture-based hand tracking method whose

complexity is sufficiently low to achieve real-time estimation while still accurately estimates hand position. The results show that the system can achieve an accuracy of more than 92% within the first 50% of gestures, and a median tracking error of less than 6.5cm.

3. Study the feasibility of fine-grained motion sensing on battery-less wearables. Due to the lack of machine learning libraries on the low-power micro controller used in the prototype, the energy consumption of the system is modeled using synthetic data based on the data-sheets of components and complexities of sensing models (e.g, HMM); whereas the evaluation of accuracy is based on the appropriate down-sampling of real-world data. Based on the analyses and prototype validation of the framework, this study examines the feasibility of real-time fine-grained motion sensing with WiFi-powered battery-less devices. Later I shall show that robust and early gesture detection using *WiWear* is achievable. When combined with other energy sources (e.g, light, motion, vibration), many additional forms of sensing context (e.g, heart-rate, skin conductance) can be supported by the battery-less wearables. In Chapter 5, I provide feasibility analyses on early gesture recognition and tracking using *WiWear* wearables. The numerical analyses show that our early gesture recognition and tracking is robust against varying sampling rates. The gesture recognition accuracy slightly degrades down to 90% when sampling rate decreases from 100Hz to 25Hz (to reduce energy consumption). The analyses also suggest that the proposed framework can support a *WiWear* wearable with an operational life time of more than 2 hours (at a sampling rate of 25Hz) with a continuous average harvested power of $90\mu W$. Our gesture-based hand tracking is significantly more light-weight than state-of-the-art hand tracking method [83] which results in a superiority in operational life time if deployed on a *WiWear* wearable.

Chapter 2

Literature Review

Though real-time motion sensing on battery-less wearable for indoor environment has not been possible in the past, recent advances in both wearable sensing and energy harvesting as well as wireless standards have enabled the *WiWear* vision. In this Chapter, I shall review the existing technologies in the literature which collectively enable *WiWear* vision.

2.1 Battery-less and Energy Harvesting

There has been a wide variety of related work in the broad areas of energy harvesting, including WiFi/RF energy harvesting, low-power wearable design, and WiFi beamforming. The maturity of WiFi (in more general–Wireless) communications has facilitated many novel applications such as activity recognition and indoor localization using WiFi signal.

2.1.1 Energy Harvesting for Client Devices

There is significant prior work on energy harvesting for wearable / embedded devices using light, kinetic energy, thermal gradients etc. Ambient and solar lighting generally provides the highest amount of harvested power as demonstrated by Heliomotes [79] to power embedded devices and Hande et. al [56] to

power indoor APs. Kinetic energy is another popular energy harvesting source that can use body movements (e.g. EnergyBug [106]) and walking (e.g. SolePower [13]) to power ultra-low-power body sensors. Energy from body movements can also be harvested in another form of piezoelectric [139] which is generated by the deformation of specific materials. Other energy harvesting modalities such as electric field variation and heat have also been studied. Electric field variation from a touch screen can be converted into usable energy by a contacting device to transmit identification tokens [90]. Thermal energy harvesting (e.g., Thermes [40] and [133]) uses temperature gradients to generate an electrical charge. More recent work, such as Flicker [57], provide a platform for rapid prototyping of energy harvesting-based sensors. Our work is complementary to these prior methods and can be (a) used to operate higher power devices, and b) deployed in environments (e.g. dark warehouses) where prior methods would not work.

Beside conventional energy harvesting techniques, harvesting energy from wireless transmissions has also been studied and usually requires custom-designed hardware for the goal of charging RFID tags and devices – with WISP [107] being a very well known example that is used to power a variety of sensors. PoWiFi [119] is the work closest in spirit, and the precursor, to our approach. PoWiFi modifies AP firmware to transmit 'power packets' (without beamforming) on multiple free channels simultaneously, and harvests such RF energy, across multiple channels, using a matched filter on the receiver. Such WiFi power harvesting is used to operate low power embedded sensors at distances of up to 20ft, but with relatively low duty cycles (e.g., a camera image once every 20 mins). Most recently, PowerBall [51] has utilized careful phase synchronization across a large number (24) ceiling-mounted RF transmitters to deliver wireless power to specific locations, enabling the harvesting of around 600μW by *static* receivers within a 20X20m^2 area. However, focusing energy to only one point is not always good, especially, when there are multiple de-

vices around the AP. Indeed, using beamforming to increase energy harvesting has been studied via simulations by Huang et. al [59] and Liu et. al [81]. I believe that *WiWear* is the first prototype to utilize directional WiFi transmissions from a single AP, together with a motion-triggered wearable sensing platform, to support human activity sensing.

2.1.2 WiFi-based indoor localization

WiWear requires accurate tracking of a wearable, potentially mobile, device, to perform accurate beamforming to relieve sufficient RF energy, so localizing a device is a required step to enable efficient beamforming.

Indoor localization has been studied for many years. Very first WiFi-based indoor localization systems exploited signal strength (RSSI) [29] as a proxy for location as signal strength is usually weaker at longer distance from the signal source. Due to multipath problem and occlusion problem, this technique is not very accurate. Channel State Information (CSI) provide richer information compared to RSSI, such as amplitude and phase of all sub-carriers, and thus has been exploited for localization [134]. Recently, the emerge of multi-antenna WiFi APs, which are used for MIMO (multiple-input and multiple-output), has enabled a more accurate indoor localization using multi-antenna APs. Prior work, such as ArrayTrack [132] and Chronos [125] have shown how to leverage active client RF transmissions, coupled with precise AoA computations to very precisely locate the client. We use similar methods in *WiWear* to detect direction of *WiWear* wearables. Device-free localization approaches such as Bharadia et. al [33], Jain et. al [61], and IndoTrack [77] were also considered. But they are not robust enough for deployment in environments with multiple human occupants.

2.1.3 WiFi-based gesture and activity recognition

The application of WiFi is not limited to communication and localization, but recently WiFi has been used for device-free gesture and activity recognition. WiSee [100] used the Doppler effect to extract the relative velocity of a moving object reflecting WiFi signal (e.g. our hand). As each gesture has a relatively different pattern of Doppler profile, a system can classify different gestures by analysing these pattern. with a relatively similar mechanism [123] use CSI instead of Doppler velocity to recognize gesture using reflective signal. In principle, one can also extract Doppler velocity from CSI. To recognize gestures using this method, a system need energy-hungry sophisticated components to extract Doppler velocity. To enable ultra low-power gesture recognition using wire less signal, AllSee [66] proposed an exciting device which can passively extract envelope of ambient wireless signal (e.g, WiFi or TV). By using a simple analog circuit to filter the ambient signal, the envelope of the wave is recorded. When a user moves the device, the envelope of the signal changes correspondingly. These patterns are then used to classify gestures. To further improve the energy consumption, AllSee designed an analog circuit to compare the envelope and achieve several μW power consumption. WiGest [23] is fairly similar to AllSee in the way that it uses changes pattern of signal strength to recognize gestures of object around a mobile device. However, WiGest does not require a user to hold any device.

2.1.4 Battery-less gestural sensing on wearable devices

Along with the exploration of new energy-harvesting techniques for powering wearable devices, gestural sensing for low-power and battery-less wearable devices has been increasingly studied. Gummeson et al. proposed and developed a battery-less ring which can track finger gestures on a surface [55]. The ring uses a microphone to detect the period the finger contacts a surface, and then uses an accelerometer to recognize specific strokes the user makes on the sur-

face. To power the ring, a coil is used to harvest the energy from an NFC reader in a mobile phone when the user holds the phone. As this technique is inductive coupling, it works in a range of 1-2 cm centimeters only [74]. With limited amount of energy, the communication between the wearable device and the host device needs to be optimized. Carvalho et al. [42] proposed an mechanism to reduce the amount of BLE (Bluetooth Low Energy) communication by transmitting only the data which is sufficiently different from the previously transmitted data. To further reduce the energy overhead of gestural sensing, the idea of using the pattern of harvested energy to recognize gestures or activities without spending energy on real sensors has been studied. Li et al. [78] proposed to use voltage patterns of an array of solar cells to recognize touch gestures when a user touch or move a finger on the cells. Lan et al. [72] explored the use of voltage pattern of a kinetic energy harvester to recognize locomotion modes (e.g, standing, walking). However, this sensing modality does not provide sufficient information for fine-grain motion sensing (e.g, hand tracking).

Table 2.1 summarizes and compares key studies which are closely related to the innovations in our method of energy transfer and harvest using WiFi signals.

2.2 Gesture and Activity Recognition

Gesture recognition is, in general, an extensively studied area. Similarly, there has been a spurt of recent activity investigating the use of smartwatches and other wearable devices to enable gestural interfaces. Given the large volume of research conducted on this topic, we focus primarily on the four aspects most relevant to our goals–(a) Effects of latency on interactivity, (b) namely inertial-sensing based gesture recognition, (c) real-time gesture recognition, and (d) 3-D hand movement tracking

Table 2.1: Summary of key related works in comparison with our technique of battery-less motion sensing wearable.

Study	Application	Approach	Advantages	Limitations
[56]	Sensor network router	Indoor solar energy harvesting	Enable near perpetual operation	Deploy close to a light bulb, large harvesters
[106]	Energy-harvesting toy for kids	Kinetic & Piezo energy harvesting	Generate enough energy to light a lamp	Need to work out to generate energy
[41]	Battery-less metering of water and heat in home	Harvest energy from the heat generated by the appliance	Can harvest up to 260μW at a temperature difference of 21$^{\circ}C$	Need to be attached to a heat source such as hot water pipe
[89]	Battery-less camera	Energy harvesting from RFID signal	Transfer enough energy to operate a low-power camera	Low frame-rate (1 frame per minute) at 1.5m from an RFID reader. RFID readers are not ubiquitously available
[119]	Battery-less sensors	Energy harvesting from WiFi signals	WiFi is highly available in urban area. Harvest sufficient energy for temperature sensors and low-power camera	Harvested energy is low (10μW at 3m). Lack of support for moving devices
[51]	Wireless energy transfer to one point	Focus RFID signal into one position in a room	Transfer up to 600μW to any point in a $20m \times 20m$ room	Continuous feedback of channel information drains energy of the device. Dense deployment of RFID transceivers. RFID is not ubiquitously available
Our technique	Battery-less fine-grained motion sensing	Energy harvesting from beam-formed WiFi signals	Apply angle-of-arrival & beamforming to boost energy density. Deliver 30μW at 3m from the AP. Support motion sensing on wearable	Lack of multi-AP support.

2.2.1 Effects of latency on interactivity

Latency has long been considered as a key factor which can make human performance degrade severely. In a study [85], Mackenzie conducted an experiment to evaluate the effects of latency to human performance. The "performance" in this study was the ability of a user to control a computer mouse to hit a target boundary on a computer screen. By introducing additional latency to the mouse movement, Mackenzie showed that user performance is not affected with latency up to 25ms, but at 75ms and 225ms, the performance was degraded with an amount of 36% and 214% correspondingly compared to the 8.3ms latency. Though later, Cheshire proposed a latency of 100ms as an acceptable threshold for interactivity [45], it was not clear whether that threshold is appropriate for any type of interaction. Claypool later explored the latency threshold for different types of online games [46]. Claypool showed that different game genres have different thresholds, and the First-Person-Shooting genre (which is highly interactive) should not have a latency more than 100ms. However, for other types of interactive applications such as sport (e.g. Table Tennis), the action speed could be much higher (up to 120 Table Tennis strokes per minute [142]), is this 100ms threshold still suitable? Later I shall show our experiment to validate the latency threshold in Table Tennis game.

2.2.2 Inertial-sensing based gesture and activity recognition

One of the common applications of gesture recognition is touchscreen-based gesture input [131, 31], where patterns of touch trajectories are classified into different input commands. Like our problem, touch screen-based recognition also needs to be near-real time, to support interactive use. Touch screen based gesture recognition and inertial-sensing based gesture recognition can both be seen as general problems of pattern recognition. However, the inertial-sensing based approach need to cope with a more difficult segmentation problem because the system does not know when a gesture starts.

Sensor-based motion gesture recognition can be used to enable detection and analysis of a variety of gesture-driven lifestyle activities. Typical studies include eating activity recognition [121, 112] or smoking recognition [94]. In this class of application, activity recognition is another potential technique such as locomotion mode recognition [102] which can provide context information of what a user is doing (e.g, walking, jogging). Higher level activity was also studied such as shopping activity recognition to analyze shopping behaviour of shoppers [111]. One of the problem in this type of application is the accuracy in uncontrolled environments because there can be many ambiguous hand motion that could be misclassified as gestures. To improve the accuracy of gesture recognition, a technique called segmentation is also applied. Segmentation selects potential segments of sensor data to classify, and thus reduces the false positive rate, but it may increase the false negative rate if the segmentation id not carefully crafted. Park et al. [96] propose adaptive threshold based segmentation which can change the threshold of acceleration based on mobility situation. However, this method requires the system to wait for a gesture to finish to avoid segmentation false. So the accuracy of the system depends on both classifier accuracy and segmentation accuracy. Furthermore, this technique is not applicable to applications that require early detection.

Recently, wearable-based sensing has also been explored to detect a variety of sports-related gestures. SwimMaster [28] and SwimCoach [86] explored the use of mobile and wearable sensors to support swimming training. Blank et al. [35, 39] recently studied the classification of Table Tennis strokes. Instead of using smartwatch, they instrumented a racket with sensors accelerometer and gyroscope to capture the movement of the wrist and the rotation of the racket. This technique enabled the classifier to be able to differentiate advanced stroke with spin. However, the user has to have an instrumented racket to play Table Tennis. Moreover, they evaluate the system using only the different strokes without non-gesture presented. Therefore it is difficult to confirm if it can be

generalized to real game situations.

2.2.3 Real-time gesture recognition

Early detection of gesture is an interesting and open problem. Recently, researchers from NVIDIA Molchanov et al. [88] apply state-of-the-art machine learning techniques including both Convolutional Neural Network and Recurrent Neural Network to recognize hand gesture online from videos. They aim to recognize gesture with *negative lag*–i.e., before the gesture has ended. In this technique, the system needs to use a sliding window to scan through the data stream smoothly (frame by frame) to detect gesture as soon as possible, so it is computationally expensive.

Bevilacqua et al. [32] tackled the problem of recognizing the progressive evolution of gestures. However, they use the conventional HMM scheme on *phrases* of sensor readings, so the processing time is very high for long *phrases*, and is not suitable for applications requiring real-time recognition. Lee et al. [76] introduce a method which can recognize gestures from video without using any sliding window or segmentation. They introduce a loop-back transition to the *start* state. However, this *start* state does not generate any observation, so it is used to restart the model. Though this method does not require segmentation, it still requires the system to wait for a while after the gesture ends to confirm the gesture and restart the Viterbi algorithm.

2.2.4 3-D hand movement tracking

Originally, hand tracking or arm tracking was done using camera [128] or camera with depth sensor [67] to provide 3-D information of user hands. However, using camera also poses certain problems such as obtrusiveness, set-up effort requirement. Inertial sensor is less obvious for tracking problems, but it is less obtrusive and easy to deploy. Inertial sensors has been used to augment 2-D positional prediction on a touch screen where several touch positions were used

as reference samples to predict touch position (up to 99ms into the future) to reduce the touch latency [75]. 3-D inertial arm tracking is more challenging as there is no explicit reference point, and the probable arm position space is much larger. Early studies focused on using inertial sensors for orientation estimation of sensors and thus body parts [53, 84, 26]. To address the accumulated drift caused by gyroscope, a magnetometer is usually used to correct the orientation estimation using the geomagnetic field [104, 130]. By deploying several inertial sensors at lower and upper arm, the arm orientation can be estimated. When combining the arm orientation and the arm length, a motion tracking system can infer positions of arm joints (e.g, elbow, wrist) [114, 130] or even full-body positional tracking with more sensors [103]. These techniques can usually provide real-time positional arm tracking (an Android device can compute the device orientation with a high update rate which can easily exceed 100Hz), but it needs several sensors to be worn at different body parts. Recently, arm tracking using inertial sensors in a single wrist worn device has been studied. Shen et al. [113] propose ArmTrak, a technique which can compute full hand trajectories. They achieve it by modeling the space around the user as point clouds, then particle filtering algorithm is used to estimate hand position. This technique requires powerful hardware to run and imposes significant processing delay (a 1-second trajectory incurs about 2-seconds of processing delay). More recently, Liu et al. proposed ArmTroi [83] as a method to address the high complexity of ArmTrak so that it can track estimate arm position and posture in real-time. It addressed the problem in ArmTrak by gradually filter out improbable states in a hierarchical manner. The technique, indeed, improves the processing time when it can estimate arm position in "real-time"; however, the processing time is still higher than 150ms which is unlikely to be applied to highly interactive application like Table Tennis training.

Table 2.2 summarizes and compares key studies which are closely related to the innovations in our method of early gesture recognition and hand tracking.

Table 2.2: Summary of key related works in comparison with our technique of early gesture recognition and tracking.

Study	Application	Sensors	Rate	Approach	Advantages	Limitations
[94]	Smoking gesture detection	Accelerometer, Gyroscope, Magnetometer	50Hz	Using 9-axis IMU sensor in a smartwatch to recognize hand-to-mouth primitive gestures and and smoking sessions	Accurate smoking recognition (95.7% accuracy)	No support for low-latency recognition
[121]	Eating gesture detection	Accelerometer	25Hz	Using 3-axis accelerometer in smartwatch for eating gesture recognition	Experiments in both lab settings and In-the-wild settings	No support for low-latency recognition
[96]	Gesture recognition for games	Accelerometer, Gyroscope	40Hz	Propose new segmentation approach to reduce the use of energy-hungry Gyroscope	Adaptive segmentation reduces energy consumption	No support for low-latency recognition.
[35]	Table Tennis gesture classification	Accelerometer, Gyroscope	1000Hz	Using a 6-axis IMU sensor attached inside a Table Tenis bat to classify different Table Tennis strokes	High accuracy (96.7%). Bat-mounted sensors can capture subtle hand movements	No *null* class. No support for low-latency recognition
[113]	Hand tracking	Accelerometer, Gyroscope, Magnetometer	200Hz	Model each possible elbow location as HMM state and use Viterbi algorithm to estimate the most probable location	Accurately estimate elbow and wrist position	High complexity, long processing time
Our technique	Gesture recognition & hand tracking	Accelerometer, Gyroscope, Magnetometer	Down to 25Hz	Avoid segmentation and evaluate samples continuously for early recognition. Use gesture states for hand tracking	Accurately recognize gestures early. Accurately estimate wrist position during gestures	Only produce accurate hand position during gestures of interest

Chapter 3

Battery-less Motion Sensing Device

Energy remains perhaps the greatest challenge in the pervasive deployment of either wearable devices for activity sensing (e.g. eating [122], smoking [95], or stress levels [49]) or embedded devices for environmental sensing (e.g., [40]). In particular, sensors such as accelerometers or gyroscopes simply consume too much energy to operate continuously without either a dedicated power source or a large battery. However, using battery power introduces three distinct disadvantages: (i) frequent recharging may simply be cumbersome or impractical–e.g., for wearable-based monitoring of elderly health at home; (ii) also, high-density storage batteries give rise to *leakage* concerns and hazards, especially when the sensors are deployed in volume and out of sight (e.g., in industrial IoT settings); (iii) in certain scenarios (e.g., when attached to the limbs of elite athletes or rehab patients with weak muscles), the added battery weight limits the imperceptibility of the wearable device and interferes with the application's intended goals.

To overcome these disadvantages, many solutions using renewable energy harvesting capabilities have been proposed–such as ambient light [56], temperature gradients [41] and kinetic energy [106]. Each such technique is innovative, but has its own limitations–e.g., ambient light cannot be used for sensors mounted in poorly lit or occluded locations (e.g., in a dark warehouse or on

occluded body locations).

In this chapter, I demonstrate the practical feasibility of using WiFi-compatible packets transmitted by a multi-antenna WiFi AP (access point) to power a wearable device with a relatively *high-power sensor*–an accelerometer (To be clear, WiFi-based energy transfer need not be the sole form of energy harvesting (even though it has certain advantages), but can co-exist with, or augment, other harvesting techniques). Wireless charging, itself, is not novel, but current solutions require either close proximity (3-5cm) to the transmitting power source (near field wireless charging, e.g., the Qi [82] standard based on magnetic induction used by modern high-end phones, which also requires precise alignment between the transmitter and receiver), or can only charge ultra-low power passive RFID tags [135] at longer ranges (far field wireless charging). More recently, PoWiFi [119] has demonstrated the use of WiFi, using multiple channels simultaneously, to power an ultra-low power wearable (with a temperature or camera sensor), with low duty cycles, while contemporaneous with our work, Energy-Ball [51] has shown how a grid of ceiling-mounted transmitters (working at 915MHz, whose propagation loss is much lower than the 2.4GHz WiFi channel) can collaboratively deliver high wireless power (as high as 0.6 mW in an 20m$\times$20m instrumented laboratory using 24 transmitters) to such tags.

In this chapter, we pose the following two *research questions*: (a) how to increase the harvested WiFi power to much higher levels (O(100μW)), even on a single channel, on an embedded device, at a much greater distance than had been previously possible? and (b) how can we perform continual gesture tracking from a batteryless, accelerometer-equipped, wearable sensing device?

We then propose a solution, called *WiWear*, which uses beam-formed transmissions, by a multi-antenna AP, of WiFi "power packets" (transmissions performed explicitly to transfer RF energy) to deliver bursts of directed WiFi energy to a client device. To point the beam towards the client, *WiWear* utilizes AoA

(angle-of-arrival) estimation techniques [132]. These AP-side techniques are paired with novel energy-conserving features on the wearable device, which activates its communication and sensing components intelligently and selectively, to help capture only key events. I also design & implement an intermittently-triggered wearable that utilizes WiFi harvesting to power a *relatively high-power* inertial sensor used in various gesture-tracking applications. To assess the feasibility of our *WiWear* solution, I numerically analyse the operations of the system under a variety of conditions. I then validate the viability of *WiWear*by utilizing multiple controlled and real-world studies.

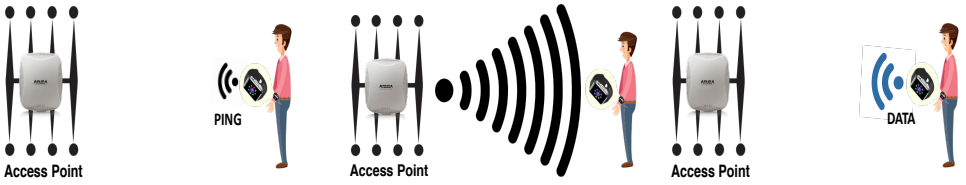


Figure 3.1: 5-step model of *WiWear* architecture. a) Step1: The wearable sends a ping packet when triggered by gestures. Step2: The AP receives ping packets and estimates AoA of the device. b) Step3: The AP sends beam-formed energy packets toward the device. Step4: The device harvests the energy from energy packets and stores it in a super-capacitor. c) Step5: the device uses the harvested energy to record sensory data, store it locally and transmit the data back to the server once available.

3.1 System Overview

In this section, I present the overall functional architecture of *WiWear*, detailing the various system-level components needed to deliver sufficiently high WiFi-based energy to stationary or mobile devices. (The detailed design of the *WiWear* wearable and AP is described later, in Sections 4.2 and 3.2.2.)

Figure 3.1 shows the overall flow of *WiWear*. In this system, the wearable or embedded device (the 'client') transmits an omni-directional 'ping' message when triggered by significant hand movements (Step 1). A WiFi AP computes the AoA (angle of arrival) of such a 'ping' message and thereby establishes the client's relative angular orientation (Step 2). The WiFi AP then transmits

electronic beamformed energy packets, delivering a more concentrated dose of RF energy towards the client device (Step 3). The client device utilizes a passive RF energy-harvesting circuit to convert this RF energy into an electrical current, storing the resulting energy in a super-capacitor (Step 4). This supercapacitor thus acts as a nano-battery, providing the transient power needed to activate the client's sensing (an accelerometer in our implementation) and communication modules when needed (Step 5). I shall show that the harvested RF energy, while two-orders of magnitude higher than prior systems, is still insufficient to power the (sensing, communication) modules continuously. Accordingly, the client device (a wrist-worn "wearable" prototype in our implementation) must employ a set of smart *activation* strategies, turning on its sensing and communication components intermittently.

3.1.1 Beamforming Technique

With the adoption of MIMO technologies in the latest 802.11n and 802.11ac WiFi standards, WiFi APs on the market are now equipped with multiple antennas: 4-antenna APs are quite commonplace, with 6&8 antenna products also becoming increasingly available[1]. The availability of such an antenna array provides us an opportunity to perform beamforming to achieve significantly more efficient power transfer. Beamforming, which is traditionally used to improve the reliability of data transfer, involves the careful control of the amplitude and phase of each antenna's transmission, so that they constructively add up in the target direction. The *beamwidth* is closely related to the number of antennas employed for beamforming: theoretically, the larger number of antennas, the thinner the beam we can achieve and thus, higher the concentration of RF power at a specific location. (Figure 3.2 shows the beamwidths that I obtained in our lab, using 4 and 8 antenna arrays.)

[1]For example, the Aruba 320 series APs (http://www.arubanetworks.com/assets/ds/DS_AP320Series.pdf)

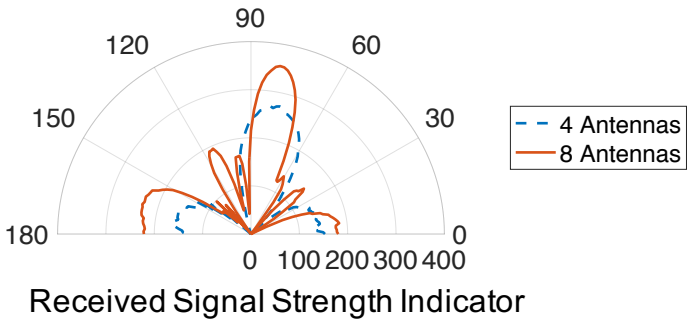


Figure 3.2: Beamwidth Observed in Practice (4—8 Antenna Array)

3.1.2 Locating the Client Device

For beamformed energy transfer to be effective, the WiFi AP needs to know the location of the client device–more specifically, the *angular direction* of the client, *relative to the AP's own location*. To compute this, the WiFi AP utilizes its antenna array to determine the AoA of any wireless transmissions from the client device. The key principle for such angle/direction estimation is that the same signal propagates different amounts of distances to reach, and thus results in slight changes in the signal phase across, different antenna elements. We employ the state-of-the-art MUSIC algorithm [110] (which has been shown in [132] to estimate AoA with errors $\leq 10 - 15°$) to perform such AoA estimation. Note also that such AoA estimation is needed only when mobile/wearable devices *move*; it is unnecessary for scenarios where the devices are static.

3.1.3 Transmission & Sensing on the Client

Each client device harvests the transmitted RF energy, stores it to cover transient demand and utilizes such stored energy to perform its necessary sensing and communication tasks. The client transfers such data only periodically (using energy-efficient bursts) to the backend/cloud infrastructure. In this prototype only, due to the limitations of the hardware, the real-time transmission is not supported yet (to be clear, it can transmit data to the AP in real-time, but at an exceptionally high cost). We leave it to the future work when we integrate new hardware to support efficient real-time transmissions. Moreover, the need to

make the client device (e.g., the wrist-worn wearable) simple and cheap implies that the client's transmissions are omni-directional. The client transmissions have two distinct uses: (i) to transfer the collected sensor data to the backend, and (ii) to provide the 'ping' packets needed by the AP to estimate the client's directionality.

3.1.4 Assumptions on System Design

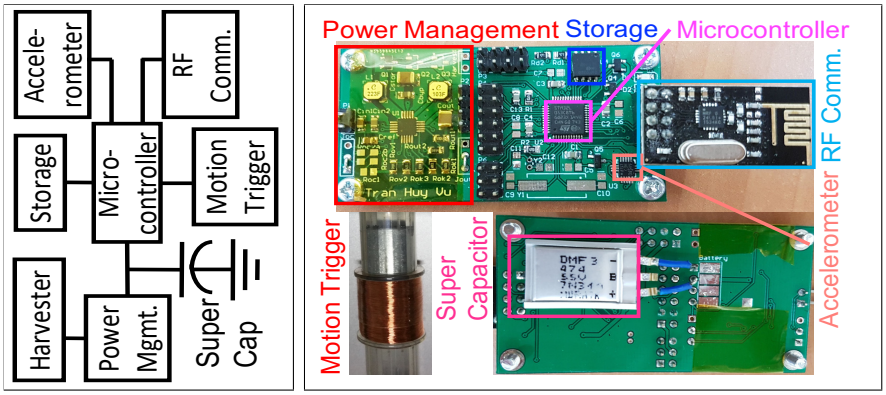
The *WiWear* system is proposed based on the following assumptions:

- *No obstacle between the WiFi AP and the wearable device:* Obstacle is an object which reflects or absorbs most of the RF signal (for example, a piece of paper is not an obstacle to RF signal). If there is an obstacle (e.g, a metal object) between the AP and the wearable device, the estimated angle will be the angle of a reflecting object, and the beam-formed "power" packets will be absorbed or scattered by the obstacle.

- *No collision of packets from the wearable device and other WiFi devices:* Though a *WiWear* AP transmits "power" packets in a coordinated manner (the AP is WiFi compatible), the RF module in the current wearable prototype does not support CSMA, so it transmits packets without waiting for an "idle" channel state. If other devices (in proximity) also transmit packets at the same time (using the same frequency range of the AP), the data will be corrupted. However, in empirical studies later, we have shown that *WiWear* can achieve reliability by repeating the transmission 15 times, even when the WiFi AP is operating with a data throughput utilization of 88% (48Mbps UDP broadcast). The experimental results presented here do not capture such possible collisions, and can thus be viewed as representative of having only a single active WiWear device.

- *An energy storage (super-capacitor) for high-power burst operation:* The *WiWear* wearable is assumed to harvest energy continually, whereas the

energy is spent in bursty fashion–e.g., only when the accelerometer sensor is active, or when the collected data is being transmitted back to the AP. Because the device's instantaneous power consumption during such active periods is much higher than the harvested power, the design assumes the presence of a short-term energy storage mechanism (a supercapacitor) that helps tide over such transient, instantaneous power deficits. The super-capacitor needs to be charged to a certain voltage, so it needs a certain amount of initial power. This can be done by transmitting "power" packets at a certain angle, and placing the device at the same angle near the AP until it achieves a working voltage threshold. After the initial charge, harvesting and using energy can happen at the same time, but the sensing pipeline (e.g, micro-controller, sensors, RF transceiver) must operate intermittently so that the harvester can accumulate sufficient energy.



(a) (b)

Figure 3.3: a) Component-level diagram. b) Wearable Implementation.

3.2 A *WiWear* System Prototype

3.2.1 The *WiWear* Client Device

We now describe the design & implementation of our RF energy harvesting based wearable device, which includes an accelerometer sensor that can help track an individual's movement and gestures. Figure 3.3a illustrates the overall component-level design of the wearable device, which contains a few key

components: an RF-energy harvester, a low-power microcontroller, the low-power accelerometer sensor, a storage unit, a wireless communication interface, a supercapacitor (to provide transient energy storage) and a power management module. Figure 3.3b shows the implementation on a PCB.

The RF Energy Harvester

The RF harvester works by converting the received wireless transmissions into an output voltage. In our current effort, we do not focus on developing the "best harvester", but instead on demonstrating the overall viability of *WiWear*. Accordingly, we implement the harvester (illustrated in Figure 3.4) on a commonplace prototype PCB (FR4 material). The harvester includes an LC network, followed by a rectifier. We hand-tune the inductor (approximately 1 round of wire) until the resonant voltage is highest on the WiFi 802.11b channel 1 (the channel used by the WiFi AP for transmitting "power packets" in our study). However, the instantaneous output voltage usually fluctuates significantly with slight movements of the wearable, implying that it is not stable enough to operate the wearable directly. We use a boost converter, BQ25570, which stores low voltage energy (as low as 100mV) and boosts it into a higher programmable voltage (set to 2.57V in our implementation) for common electronic devices. This output voltage is then used to operate an entire embedded system including 1 microcontroller, 1 inertial sensor, and 1 RF communication front-end.

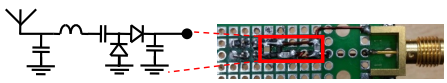


Figure 3.4: RF Harvester: FR4 PCB & hand-tuned inductor.

The Microcontroller+ Sensor

We utilize a commodity low-power microcontroller, the STM32L053 [14], which consumes 6.6 mW power at normal operation, but only 1 μW power during stop mode. In stop mode, all functions of the device are stopped, but the content of RAM is preserved. In our system, when the accelerometer records

enough data, it generates an interrupt signal to wake up the microcontroller to read the buffer. The microcontroller wakes up every 3 seconds to read 90 bytes of acceleration data from the accelerometer if the accelerometer is actually active. The wearable can store the acceleration data in a FRAM storage unit, the Cypress FM25VN10, (for a transmission burst later) or transmit the data back to the server using the RF front-end. Our device implementation uses the LIS3DHTR 3-axis accelerometer from STMicroelectronics. This low-power sensor consumes 2 μA at 1 Hz, and 6 μA at 50 Hz. According to [27], 98% of frequency spectrum power of accelerometer signals for human activities such as walking lies under 10 Hz. Accordingly, we use a sampling frequency of 10 Hz, as this proves adequate in tracking most natural gestures (in addition, gesture recognition approaches often filter out high-frequency noise).

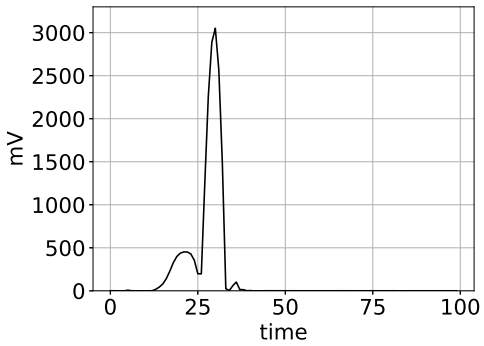


Figure 3.5: Voltage generated by motion trigger.

Zero-Energy Motion Trigger

To minimize the unnecessary energy drain of the wearable device, we adopt a triggering-based mechanism, whereby the sensor and the microcontroller are activated only *when the wearable device experiences significant motion* (e.g., when the user makes a gesture). To avoid the energy drain from such motion monitoring, we include a very simple, "zero-energy", passive, motion trigger: a coil (taken from a shake torch) with a Neodymium magnet inside. Whenever the device is subject to a significant movement, the coil generates a voltage high enough (see Figure 3.5) to trigger an external interrupt to the microcontroller, which then activates the rest of the components. This trigger also causes the

controller to generate and send out 'ping' packets, which the AP can then use to infer the client's updated AoA. Our motion trigger component is more sensitive to rotational movements but less sensitive to subtle linear motion. However, this was not a limitation in our current studies (in an office meeting room), where user gestures typically include a sufficient rotational component. There are prior studies on tiny MEMS-based motion energy harvesters [87, 136] that may provide greater linear and rotational motion sensitivity.

3.2.2 The *WiWear* AP

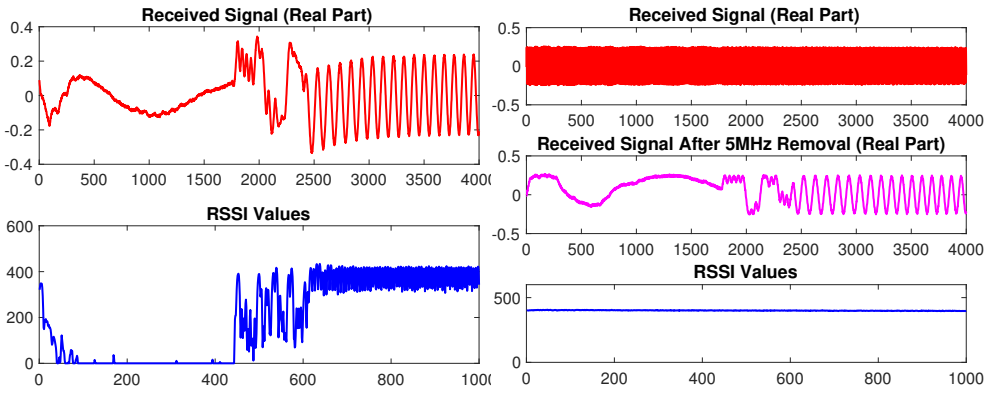
We now describe the design and implementation of our enhanced WiFi AP. To support the beamforming-based RF charging vision, the AP needs to perform the following additional functions: (i) detection of the 'ping' packets; (ii) determination of the AoA and (iii) beamformed transmission of 'power' packets. We implemented our functionality using the WARP [5] platform, which is widely used within the research community. By default, each WARP board can support a maximum of 4 antennas. To support more precise beamforming using an 8-antenna AP, we coupled the operation of 2 separate WARP boards (Figure 3.7). To enable beamforming and AoA estimation, the phase difference among the antennas (across the two boards) must be precisely calibrated, and they must capture or transmit data at *exactly* the same time. We use a CM-PLL cable to synchronize the operation of the two boards, setting one board as a master to perform all the functions of a regular 802.11 AP. The second board performs as a slave, receive packets (for transmission) via the Ethernet interface and transmitting them wirelessly when triggered by the master. To support dynamic beamforming of power packets, we insert a complex multiplier at each antenna interface whose coefficients are specified within the power packet. Though the transceivers on the WARP board support both 2.4 GHz and 5 GHz band, the reference design supports only 2.4 GHz. In newer APs, one may conceivably use the 2.4 GHz band for energy and the 5 GHz band for usual data

communication. Operating in a higher frequency band (e.g., 5 GHz) involves a tradeoff between higher path-loss, but greater possible number of antenna elements (providing narrower beams) due to the smaller wavelength and resulting inter-antenna gap. I further explore ways to tackle these open challenges in my on-going work (discussed later in Section 4.8).

Detection of Low-power GFSK 'Ping' Packets

The *WiWear* wearable uses a low power NRF24L01+ module to transmit the 'ping' packets, whenever it is subject to a significant movement. This RF module uses GFSK modulation with a maximum 2Mbps data rate. The preamble of each packet is merely 8-bits ("01010101") followed by a 3-5 byte address. This makes the packet detection by the AP much more challenging compared to usual WiFi packet detection for 2 reasons: 1) These packets are not WiFi compatible. The preamble is too short compared with a usual WiFi preamble (hundreds of symbols). 2) The signal is too narrow band, with each packet preceded by 1.5 cycles of very low frequency ($\sim$40 KHz) while the RSSI circuit in the transceiver computes the RSSI across the whole range of 20MHz bandwidth. This produces very low and unstable RSSI readings of 'ping' packets at the WARP, which must support the wider 20 MHz band of WiFi and thus generates incorrect gain values from its Automatic Gain Control (AGC). I tackle this problem by using the frequency overlap between consecutive WiFi channels (consecutive channels have 15MHz overlap, the space between the 2 center frequencies is 5MHz): the wearable transmits such packets at the next higher channel, while the WARP board is tuned to the lower channel (see Figure 3.6). Specifically, in our current settings, the RF device transmits at channel 2 (center frequency at 2417MHz) while the WARP AP uses channel 1 (center frequency at 2412MHz). As a consequence of the resulting 5 MHz shift between the transmission and reception center frequencies, the received signal bandwidth becomes wider because 5MHz will be automatically added to the

original GFSK signal (Figure 3.6b, the top plot). Therefore the transceiver on the WARP board produces a significantly more stable signal. The receiver (AP) then needs to remove the 5MHz from the received signal to restore the original narrow-band GFSK signal (Figure 3.6b, the middle plot). Figure 3.6 also shows that the received signal using overlap channel is less affected by DC offset.



(a) (b)

Figure 3.6: a) Received signal and RSSI values from nRF24L01+ device and corresponding RSSI recorded at the same channel. The RSSI is unstable and some parts become zeros. DC offset is also observed. b) Received signal of another packet (before and after applying -5MHz shift) and RSSI values using channel overlap. Much more stable signal is observed with almost no DC offset.

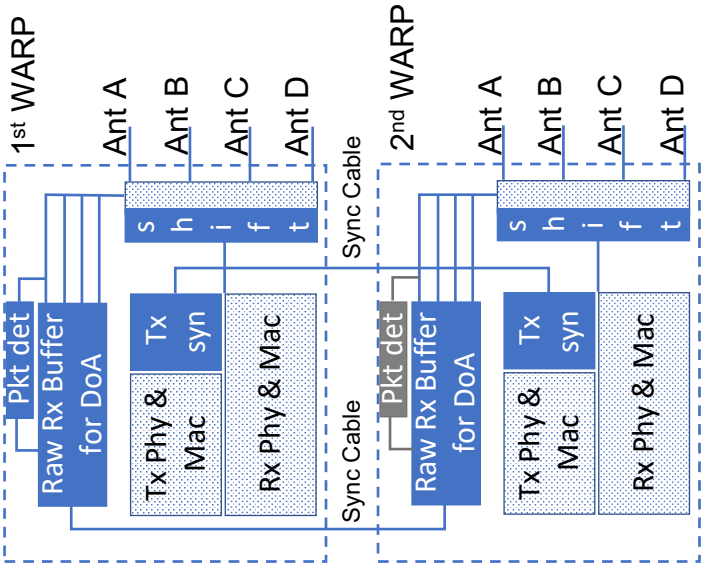


Figure 3.7: AP Modification for beamforming and AoA. The dark blue parts are our extension.

Extension for AoA Estimation

The default 802.11 reference design in WARP board supports only 1 receiver path, though the system can switch among either its 4 antennas. As AoA estimation requires simultaneously data capture from multiple antennas concurrently, I modified the design to add a circular buffer. This buffer stores the data from all 4 antennas whenever a packet (from a wearable) is detected, and can also store similar data from the 4 antennas on the slave WARP board. A control server then reads the packets in the buffers to estimate the AoA of incoming packets.

Extension for Beamforming

To support the per-antenna phase control needed for beamformed transmission, I insert a complex multiplier (whose coefficient can be controlled through a register) to each antenna output. This allows the dynamic change of the phase of any transmitted packets before it is transferred. We also need to overcome an additional challenge, in ensuring concurrency when the two WARP boards are used concurrently. If each WARP operated as an independent AP, their transmission schedules could differ, due to differences in their underlying carrier sensing. To overcome this, we disable the MAC layer of the slave board so that it will transmit a packet as soon as the master sends it a 'transmit' signal.

3.3 System numerical analysis

Before any evaluation with physical devices, it is interesting to see how the principles of the system are applied to deliver energy to harvesting wearables. At this stage, we assume that using AoA estimation and beamforming techniques (introduced in Section 3.1) the AP can detect the angle (direction) of harvesting device accurately. The AP is assumed to have 8 antennas, each can transmit signal at 20dBm or 100mW. We also assume the transmission channel conforms to *free-space-path-loss* model. Under these assumption, we can model the energy

received by a harvesting device using the following equation (See [54]):

$$E = \frac{TxPower * TxGain * RxGain * \lambda^2}{(4 * \pi * d)^2} \tag{3.1}$$

where $TxPower$ is the total transmission power of the AP antenna array. $TxGain$ and $RxGain$ are the AP antenna array and the receiver antenna gain correspondingly. λ is the wave length of the signal which is $\sim$12cm (2.4GHz). d is the distance from the AP antenna array to the receiver antenna.

This analysis will provide a sense of feasibility of the system or the delivered energy in the best case. Of course, in real implementation, many other factors may affect the delivered energy such as multi-path effect (which causes AoA inaccuracy and destructive resonance), power conversion efficiency (AC to DC), antenna direction. To better understand the affect the the design choices of the system, we assume the perfect wireless channel conditions and power conversion efficiciency. I shall show later the feasibility analysis of the system based on simulation results (using Matlab and Phased-Array toolbox) in both single and multiple-device cases.

3.3.1 Expected system energy consumption

Table 3.1: Power consumption of off-the-shelf conponents needed for a motion sensing wearable device.

Component Name	**Power Consumption**	**Condition**	**Sample**
μController	$7.5mW$	16MHz	STM32L053 [14]
Accelerometer	$15\mu W$	50Hz	LIS3DH [11]
Gyroscope	$875\mu W$	52Hz	ISM330DLC [10]
Magnetometer	$62\mu W$	20Hz	IIS2MDC [9]
RF Module	$27mW$	2Mbps	nRF24L01+ [12]
FRAM Storage	$0.75mW$	2Mbps	FM25VN10-G [8]
Super Capacitor	$16\mu W$	5V	AMK432BJ477 [7]

To understand the energy demand of wearable devices to support sensing applications, we consider several application scenarios with respect to power

consumption of off-the-shelf sensors. Table 3.1 provides the power consumption of sensors and components needed for a motion sensing wearable device. Though several components have much higher power consumption compared to others, they are not necessary to stay active all the time. For example, the Micro-Controller only needs to stay active to read the sensor data (when the sensor buffer is full) and to transmit the data back to the AP. Likewise, the RF module needs to be active only during this active data transmission phase. An intelligent system would be able to apply event based operations to trigger energy-hungry components only when it is really necessary, and thus reduce the power consumption of the entire system.

Scenario 1: Battery-less patient monitoring application

Considering a patient monitoring application where doctors want analyse a patient motion pattern (activities, gait, etc.) and energy expenditure to give appropriate consults. For this type of application, the system can use accelerometer to monitor the physical activity level of a patient [3, 108]. As the doctor needs to analyse the data for an entire day, even when the patient is sleeping, so the system needs to record accelerometer data continuously during a day. This application does not require real-time sensor streaming, so we can assume that the device transmits the data back to the AP at a specific time in a day (e.g. sleeping time). Assume the accelerometer running at 50Hz, the sensor can buffer 32 samples before the Mocro-Controller has to read the data out. During a day the sensor consumes 1.3J. The Micro-Controller needs to wake up 135000 times to read data. Each time it takes 384μs to read 32 bytes of data, thus 0.39J. It also takes the same amount of energy to copy the data to the RF module. The RF module takes 3.6 times as much as energy to transmit all the data, thus 1.4J. The FRAM module is active while data is storing and transmitting, so it takes 0.078J. Finally, to track the device to adjust the beam direction, the device needs to send a "ping" packet in every T time (assume 1 minute), so the device spends

0.019J for “ping” packets. In total, the system consumes 3.2J for an entire day. So the supply power should be at least $37\mu W$ on average (throughout a day) to support this application.

Scenario 2: Battery-less VR game controller application

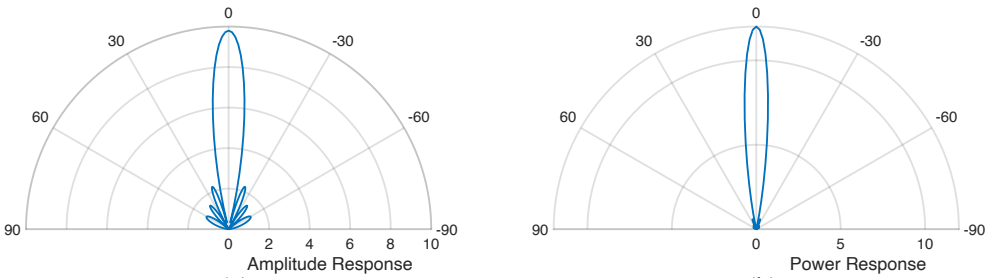
In this scenario, we analyse how much energy the system needs to support a VR-Game playing session. In this case, the controller, which could be in the form factor of a ring, is a battery-less device which can functions as a game controller. This application requires real-time streaming of sensor data. Also the system might need more than one sensor. Previous studies [113, 127] showed that early gesture recognition and hand tracking can be achieved using 3 sensors: Accelerometer, Gyroscope and Magnetometer. We assume that the system use the aforementioned 3 sensors. Of course, we cannot expect to use the energy from WiFi to play VR game continuously. Hence, we assume that the device is charged continuously throughout the day, but only gets activated at a few random time instants (when the user presses a button to activate “Game Mode”). The device use the stored energy to read the data from all sensors and stream it back to the AP. We assume the playing session is one hour long. Similar to the above patient monitoring application, the device spends 0.019J for ”ping” packets. During 1 hour, the 3 sensors (accelerometer, gyroscope and magnetometer) consume 3.4J ($952\mu W$ for 1 hour). The device wakes up 5625 times to read sensor data. Each time, it reads 96 bytes from the 3 sensors which results in 1.15ms. The total time the RF module takes to transmit 96 bytes is 1.45ms (transmitting short burst is less efficient). The device can do both data transmission and data reading concurrently, so the time for both is 1.45ms. During 1 hour, both Micro-Controller and RF module stay awake for 8156ms. In total, the energy expense is 7.17J. So the system must be able to supply at least $83\mu W$ on average (throughout the day) to support this application.

Based on these analysis of basic power demand, we perform several sim-

ulations of the system to explore the feasibility of the proposed system and techniques to enable those envisioned applications.

3.3.2 Single device scenario

Static device



(a) (b)

Figure 3.8: a) Amplitude response of 8-antenna array with one beam. b) The corresponding power response.

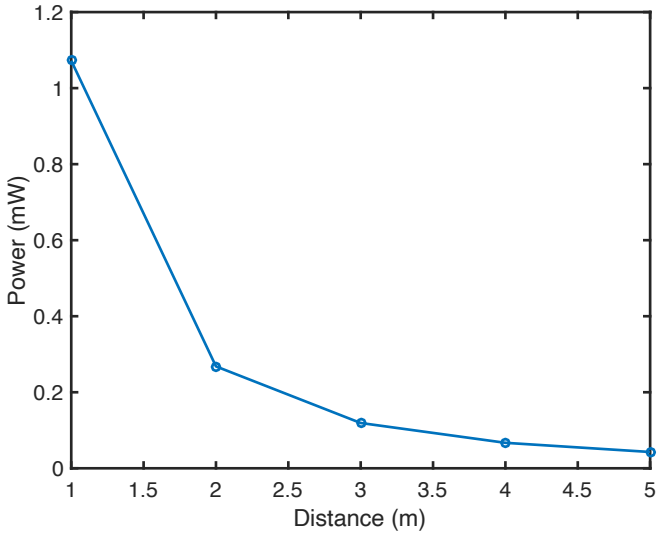


Figure 3.9: Upper bound of energy harvested at different distances.

The most simple situation is the case when there is only one harvesting device in the vacinity around the AP, and the device does not move. The AP needs to detect the device angle only once and keep boosting energy into that direction. This is equivalent to the situation when the AP knows exactly the angle of the device at every moment and changes the beam of signal toward the device instantly. This is the simplest scenario but it does not mean it is unrealistic. This can be applied for situations when a user places an device (an object with

a battery-less sensing device – a tag) at a specific position and he rarely moves it. Whenever he moves it, he recalibrates the position (by pressing a button in an administrator app). Figure 3.8.a shows the amplitude response pattern of the 8-antenna array when beam-formed into one direction, and Figure 3.8.b shows the corresponding power response. The maximum value of amplitude response is higher than 8 as we use *short dipole* antenna model for each of 8 elements. However, these values only represent how strong the signal is emitted at different directions. As shown in equation 3.1, the received power also depends on the gain of the receiver (harvester) antenna and the distance. Figure 3.9 shows upper bound of delivered energy at different distances without considering the mismatch between the estimated angle and the actual angle of the device which can be caused by both the inaccuracy of AoA algorithm and by the motion of the device. The results suggest that the harvested energy is sufficient to power a single device at practically long distance. Even at 5m, the AP still supplies sufficient power to a device (non-real-time sensing).

Moving device

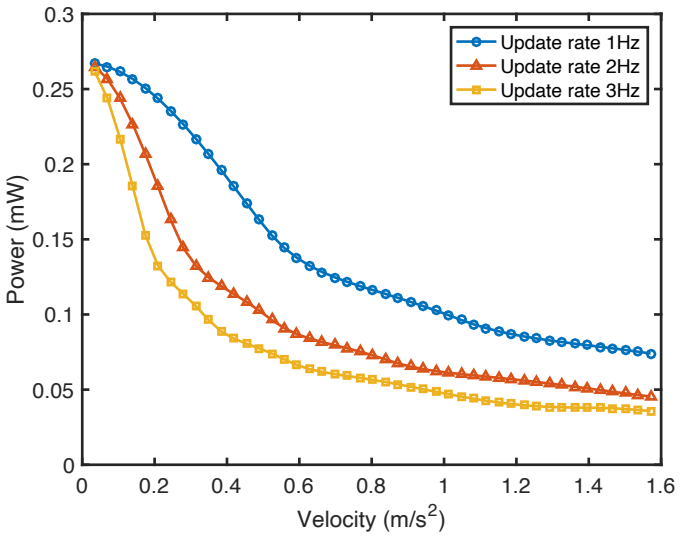


Figure 3.10: Harvested energy when the device is continuously moving at different speed.

When the device is not static, but moving, the system behaves differently as there will be displacement between the pointing direction of the AP and the

current direction of the device. Recalling the overview system (discussed in Section 3.1), the AP only updates the beam when the device sends a packet to the AP. Of course, the device cannot send a continuous wave for infinite period of time so that the AP knows exactly the angle of the device at any moment. The device needs to send a packet in every T seconds, but T cannot be too small as wireless transmission drains energy of the device. If the device sends a packet to the AP at time t_0, by the time of the next packet at time t_1, the actual angle of the device might have changed significantly already. In general, the amount of displacement changes differently in response to the velocity of the device and AoA/Beamforming update frequency. For example, a harvesting device is moving continuously with an angular velocity of v_d, the update interval is T_u, the displacement is equal to $v_d \times t, where\ t = 0\ to\ T_u$.

Figure 3.10 shows the harvested energy when the device is moving continuously at different speed from $0m/s^2$ - $1.6m/s^2$ at a distance of 2m (the device moves in a circular trajectory). Note that the usual walking speed is $1.4m/s^2$. It can be easily seen that at reasonable speed, the harvested energy is low even at an update interval of 1 second. It suggests that the system is not likely to work with objects that continuously move at speeds comparable to usual walking speed, but it could be applicable to the case of patient wheelchair whose speed is low and limited. Fortunately, people do not move continuously. Analysis of movement data at Singapore Management University, obtained using the LiveLabs testbed platform [30] showed that, on average, each person spends $\sim$92% of their time in stationary mode. For instance, an employee in an office usually stays at his desk to work for 2-3 hours before going to the toilet for a couple of minutes.

Assume a scenario where a device moves for 10 minutes in every 1.5hours (stay at a position for 1.5 hours) during 10 hours. The devices move randomly with a speed of 1.0 to $1.4m/s^2$; the distance from the device to the AP can have any value between 1m and 3m. The angle of the device is arbitrary. To

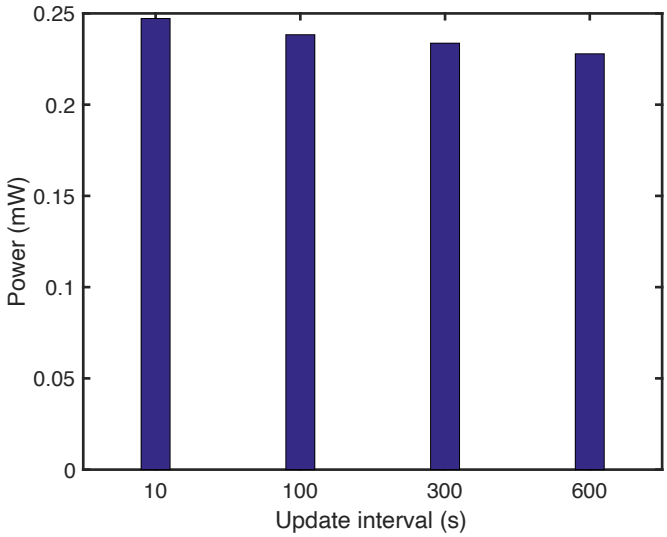


Figure 3.11: Average energy harvested by a single device during 5 random traces of intermittent ”move” and ”stay”.

create random trajectory, the device changes its moving direction randomly after every 0.1s. The estimated energy is averaged over 5 such random trajectories. Figure 3.11 shows that the update interval between 10 seconds and 10 minutes does not considerably affect the harvested energy. Of course, shorter update interval (higher update frequency) results in slightly higher harvested energy, but in real application, the system needs to spend more energy to send “ping” packets. In this specific scenario, changing the update interval from 10 seconds to 1.6 minutes results in only about $8.8\mu W$ lower harvested power. But updating the device angle at every 10 seconds incur more energy as the device needs to send “ping” packets 10 times more frequently. This simulation suggests that the system can definitely support a single motion sensing device, which can move in a range from 1m to 3m, with a reasonable moving/staying ratio of 1/10.

3.3.3 Multi-device device scenario

Static devices

Similar to the static single device scenario, the AP needs to perform beamforming only once and keep that configuration permanently. However, as can be seen in Figure 3.8a, the beam is unlikely to cover many devices (if they are sig-

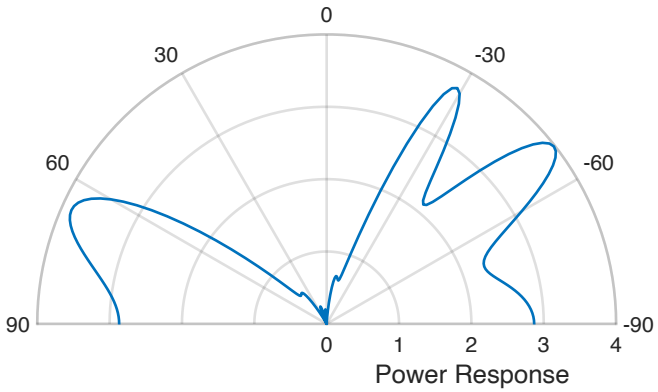


Figure 3.12: Beam pattern control (4 devices at $-60\,\text{deg}, -25\,\text{deg}, 60\,\text{deg}, 80\,\text{deg}$) .

nificantly separated). The simple approach would be time-multiplexing among the devices, but the energy received by each device is almost evenly divided by the number of devices. The question is that, should the system adopt the simple multiplexing technique or beam shaping technique (controlling the antenna radiation pattern) to cover multiple devices?

To tackle this situation, I developed an optimization algorithm for beam shaping to maximize the energy delivered to multiple devices. Conceptually, I divide the antenna array into K subarrays of size N, k^{th} subarray can point to direction $\Phi_k, k = 0..K-1$, and seek to determine the pointing direction of each sub-array so that the energy objective is maximized. We consider 2 different objective functions: (a) maximize the total energy harvested (Max-Sum) and (b) maximize the minimum harvested energy, across the devices (Max-Min). The second objective promotes fairness by ensuring that no individual device "starves" of energy.

For Uniform Linear Array (ULA), the phase of the k^{th} sub-array (and thus its pointing direction) is adjusted by adjusting the phase of each of its antennas in a linear fashion according to: $Phase(Antenna_k^n) = (K \times N + n) \times \Phi_k, n = 0..N-1$. Using this assignment, each sub-array forms one beam separately; however, if two sub-arrays point to the same direction, they effectively form a

narrower single beam. Hence, the angle-selection problem becomes:

$$\text{Max-Sum} : argmax(\Sigma_{i=0..M-1} P(\Phi_0, \Phi_1, ..., \Phi_{K-1})[\theta_i, d_i]);$$

$$\text{Max-Min} : argmax(\min_{i=0..M-1} P(\Phi_0, \Phi_1, ..., \Phi_{K-1})[\theta_i, d_i]);$$

$$\text{Multiplex} : argmax(\min_{i=0..M-1} \Sigma_{k=0..M-1} \alpha_k \times P(\theta_k, \theta_k, ..., \theta_k)[\theta_i, d_i]);$$

where Φ_k is the phase shift of the k^th antenna element, θ_i and d_i are the angle and distance of the i^{th} device correspondingly. α_k is the percentage of time the beam point to the k^th device. In the case of $multiplex$ strategy, all the sub-arrays are aligned into the same direction. $P(\Phi_0, \Phi_1, ...\Phi_{K-1})[\theta_i, d_i]$ is the power received by the i^{th} device. The power is cumputed using the equation 3.1.

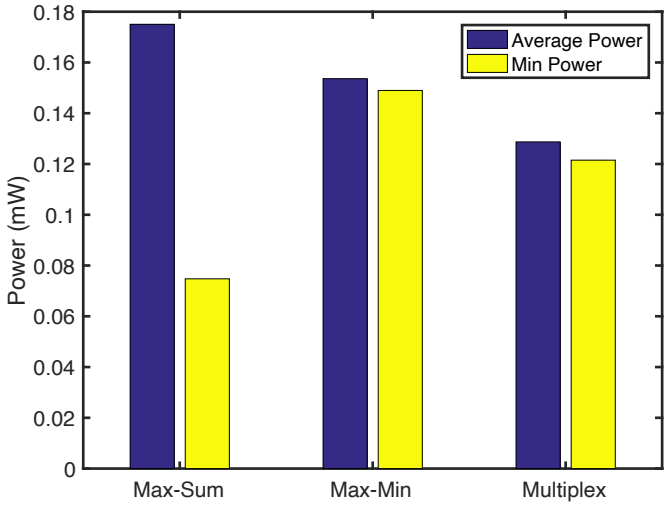


Figure 3.13: Average energy harvested by each device and average minimum harvested energy using 3 strategies (100 random positions, distance = 2m).

The amounts of harvested energy using 3 different strategies are shown in Figure 3.13. These numbers are achieved by averaging the harvested energy of a device in 100 iterations. Each iteration includes 4 devices distributed randomly around the AP at a distance of 2m. It can be seen in the figure that $max - sum$ strategy achieves the highest average power, but there could be starved devices. $max - min$ strategy achieve a bit lower average power, but the no device is starved. $multiplex$ strategy achieves lower power compared with $max - min$ strategy in both average and minimum power, though it achieves much higher

minimum power compared to $max - sum$. In practice, because of the inherent characteristic of AC-to-DC converter, if the input power is too low (e.g. input voltage $\leq$ 200mV for BQ25570 converter), the converter cannot start the conversion process. In this case, the system needs to switch to $multiplex$ strategy as the single beam is much more powerful.

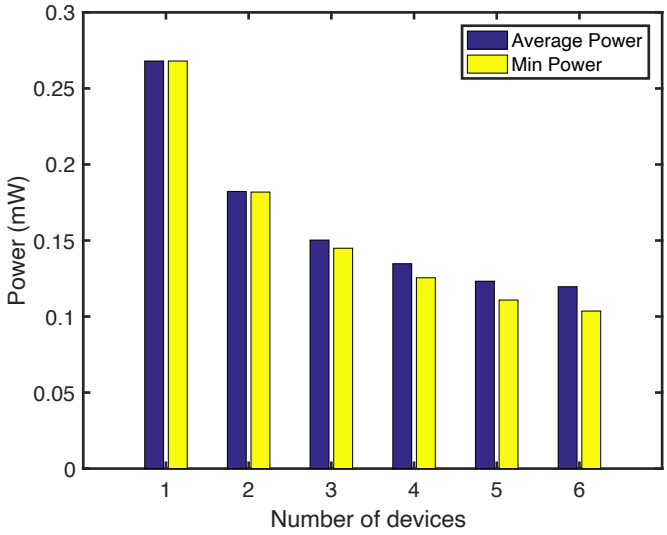


Figure 3.14: Average energy harvested by each device with different number of devices (100 random positions, distance = 2m).

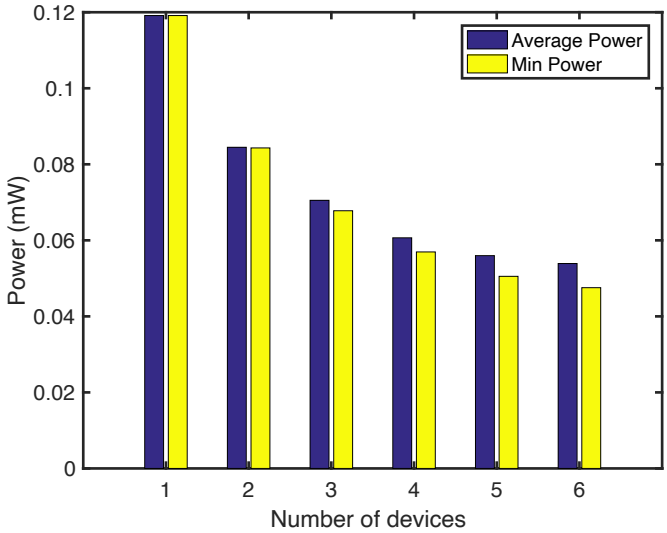


Figure 3.15: Average energy harvested by each device with different number of devices (100 random positions, distance = 3m).

Another intersting point is that the power is not simply divided by the number of devices even in the case of $multiplex$ strategy because there could be two or more devices sharing a beam, or some devices take the advantage of side lobe. Figure 3.14 shows the harvested power with different number of devices.

Even with 4 concurrent devices (at a distance of 2m), the average power of each device still reach $\sim 140\mu W$, and no device harvests less than $130\mu W$. At 3m (Figure 3.15), the average power of one device is more than $60\mu W$. With these values, it should be able to tolerate the imperfections of components in practical conditions.

Moving devices

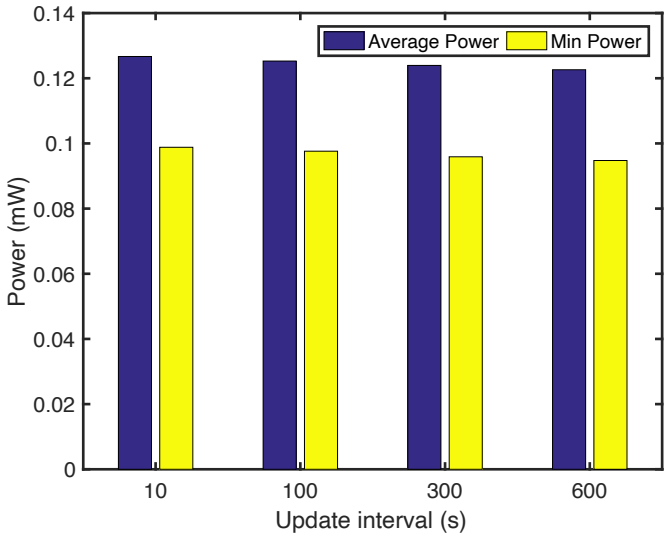


Figure 3.16: Average energy harvested by 4 devices during 5 random traces of intermittent "move" and "stay".

Beam shaping for one device usually maintains the same power gain at any angle of the device (as the AP can adjust the beam direction). In the case of multiple devices, depending on relative position of the devices, the beam pattern will be different, causing different power gain at different directions. Similar to the case of a single device, I do not target scenarios of continuous moving devices. I target scenarios where a device move and stay at a position intermittently. The harvested energy is averaged over 5 10-hour scenarios. In each scenarios, I also assume the devices spend 90 minutes staying at a position then spend 10 minutes moving randomly to other positions. Each device have a different speed of 1.0 to $1.4 m/s^2$. Figure 3.16 suggests that the update interval can be relatively high compared to the moving time without sacrificing considerable harvested power as the radiation pattern is more widespread. With considerations of the case of

a single device, the update interval of 1.6 minutes would be a good combination as it requires only 0.015J for "ping" packets throughout a day ($0.17\mu W$).

To summarize, the numerical analysis suggests that the system can support 4 devices if the devices are in the range of 3m from the AP. The harvested energy is sufficient to enable our potential applications of patient monitoring and VR-Game controller.

3.4 Performance Evaluation: Micro-Benchmarks

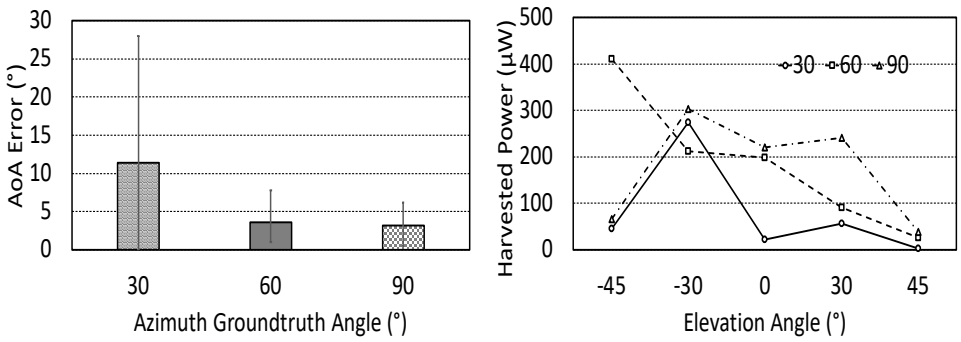
In this section, we shall study how *WiWear* works under *controlled* conditions– i.e., when the *WiWear* wearable platform is stationary, and not mounted on any real user. These micro studies help establish the performance characteristics of each individual component (e.g., AoA determination, beamformed energy harvesting) and the resulting impact on the harvester energy output, under different conditions.

3.4.1 Experiment Setup & Calibration

All our experiments were conducted in a meeting room (3.5m x 4.5m) of our university building. The WARP system was installed on a table (1.1m x 1.9m) in the middle of the room–Figure 3.21 (shown later) demonstrates the setup used. For these studies, we use the same wearable device used in user studies (Section 3.5), but we do not connect the harvester's output to the power management unit of the wearable device. The AP transmits standard 802.11 packets with different power levels, but it transmits power packets at maximum power (20dBm) per antenna; thus, *in all experiments, our total transmitted power was well within the EIRP upper bound of 800mW*. A software program running on a computer generates 1024-byte 'power' packets (UDP packets with phase coefficients of 8 antennas) *continuously* (except for the study in Section 3.4, where we intentionally varied the percentage of 'power' packets). We place the wearable

device on a tripod. For each study setting, we manually trigger the wearable device to transmit 'ping' packets, such that the AP can update its beam to point in the estimated direction of the wearable. We then record the average power of the harvester output with a 10kOhm resistive load.

3.4.2 Change in Azimuthal Orientation



(a) (b)

Figure 3.17: a) AoA Estimation Error. b) Harvested energy at different azimuth and elevation angle.

We investigate the performance of the system (both AoA and Energy Beamforming) under different angles, with the wearable placed 1 meter from the AP. In theory, the performance of the system from 0° to 90° should be similar to 180° to 90° (the front half of the *azimuth* plane). However, the beam intensity in the space below and above the antennas (i.e., for different values of the *elevation*) should be different. We measure the system performance with 3 different azimuth angles {30°, 60°, 90°} and 5 different elevation angles at {-45°, -30°, 0°, 30°, 45°}.

Figure 3.17a shows the AoA estimation error for different azimuth and elevation angles. It is known that the MUSIC algorithm becomes inaccurate as the azimuth angle approaches 0° or 180°. Indeed, we see that the AoA error is $\leq 5°$ when the azimuth angle $\geq 60°$, but reaches a median value of 12°, when the azimuth is 30°. However, 120° (30° to 150°) is indeed an unnaturally wide field of view for practical scenarios. Figure 3.17b shows the harvested energy accordingly. The results suggest that the harvested energy remains fairly high as

the elevation angle from -30° to 30°. In our office room setting, the AP is able to cover almost the entire room with an elevation angle of 30° and azimuth of 60°; within this space, the harvester is able to harness over 200 μW.

3.4.3 Energy harvesting vs. Distance

I next study how the efficiency of energy transfer diminishes with an increasing AP-wearable distance. Figure 3.18 shows the harvested energy, as the distance is varied from 1m-3m., with (azimuth=90°, elevation=0°). The results show that, even at 3m, the AP can still transfer about 33μW to the harvester. Given that our wearable with drains out only 23μW (from the built-in 220μF supercapacitor) even when it is continuously recording the accelerometer reading (without transmitting 'ping' packets), we see that *our paradigm of beam-formed WiFi energy transfer is able to support the uninterrupted operation of the* WiWear *wearable essentially* ***anywhere*** *within a standard meeting room.*

3.4.4 Energy harvesting vs. Background data

I next study how the energy transfer efficiency is affected by the need for the AP's power packet transmissions to co-exist with regular WiFi data packets. Note that our modified AP implements the 802.11 AP reference design (from MangoComm [6]), and is thus able to provide data connectivity to regular WiFi clients. We study the sensitivity of efficiency by varying the percentage of

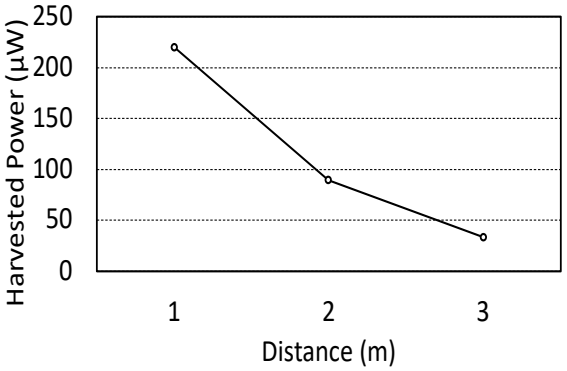


Figure 3.18: Harvested power vs. Distance

broadcasted IP (data) packets & power packets. Figure 3.19 shows that the harvested energy decreases quite linearly with the percentage of IP packets. When the AP exclusively transmits IP packets (100%), the harvested power is almost zero as the AP transmits such packets using only 1 antenna and usually at less than the highest permitted power level. At the typical utilization (20%) observed on our campus WiFi network, *WiWear* appears to be capable of harvesting 200μW, a 100-fold increase from ambient power levels.

3.4.5 Effect of Number of Antennas

We next vary the number of transmitting antennas in the WARP transmitter and study the impact on the harvested power (wearable-AP distance= 1 meter). Figure 3.20 plots the harvested power. Matching our intuition, a larger number of antennas allows the transmission beamwidth to be thinner, thereby effectively increasing the density of the delivered RF power. Interestingly, we observe that the angle at which the harvester gets the maximum energy differs a bit from the ground-truth, with the difference increasing from 5° to 20°, when the number of antennas is reduced from 8 to 2, respectively. We suspect that this is due to the inevitable errors in phase control, with the phase errors getting averaged out when the number of antennas is higher. However, in practical environments, an overly thin beam may be counterproductive as errors in AoA estimation (especially in more crowded environments) may cause the narrow RF beam to be misdirected, resulting in a very sharp drop in the power harvested.

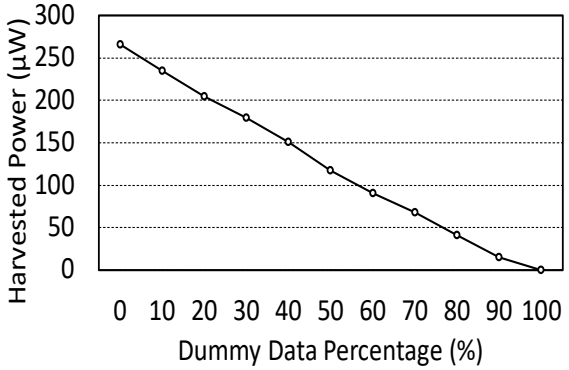


Figure 3.19: Harvested power vs. Varying 'data' traffic load

3.5 Constrained User Studies

We now evaluate the performance of the *WiWear* prototype, under constrained user studies performed in our 3.5m x 4.5m meeting room, set up to mimic a typical work environment. These studies are reviewed and approved by our Institutional Review Board (IRB-18-098-A083(918)). "Constrained" refers to the fact that the users are requested to stay within the meeting room during the study duration (30 minutes) and perform their "normal" office activities, while wearing the *WiWear* wearable device. Each user is, however, free to perform one or more activities of their choice (e.g., typing on a laptop, taking short breaks and stretching, etc.). Unlike Section 3.4, the wearable is now subject to human-specific movement and resultant changes to its performance metrics (e.g., AoA estimation error and fluctuations in harvested energy). As before, experiments are performed using an 8-antenna AP array, with the maximum total transmission output of 800mW, (*below the EIRP limit*).

We studied the behavior of 4 distinct users, each of whom was asked to initially sit at a different corner of our 1.1m x 1.9m table (see Figure 3.21) and subsequently perform their usual desk-based office chores for 30 minutes. The super-capacitor helps tide over the fluctuations in harvested power, caused due to such arm movements. We also experimented with smaller capacitors: Figure 3.22 shows the transient shortage of energy when using a small capacitor (10μF). Our studies revealed a trade-off: a larger (0.47F) capacitor can

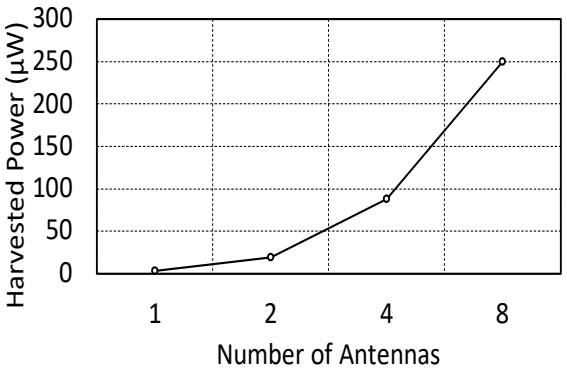


Figure 3.20: Harvested power vs. No. of antennas

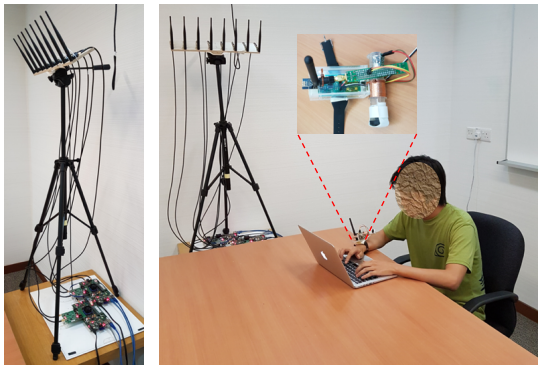


Figure 3.21: Experimental Setup: (a) Left: The AP, comprising 2 WARP boards. (b) Right: A user wearing the *WiWear* device during the study.

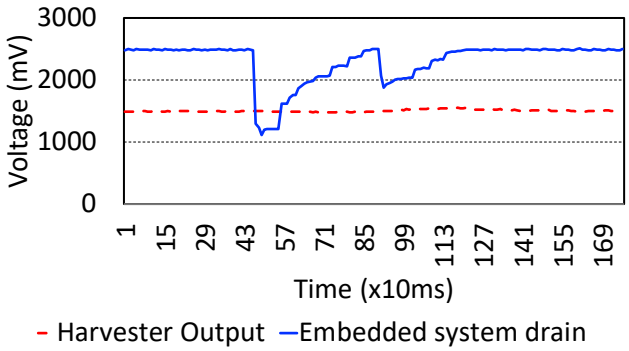


Figure 3.22: Time series of wearable voltage using 10μF small capacitor.

buffer enough energy for hours, but its leakage is higher ($\sim$13μW higher) than a smaller (220μF) capacitor, which can support the wearable operation for less than a minute. To deal with longer-lived periods of deficient harvesting (a user might cover his hands or move it to a blind spot for several minutes), we chose the 0.47F super-capacitor, which ensures that the available energy is never depleted during the experiment. The overall operation of the *WiWear* wearable is then as follows: The device usually sleeps, until the motion detector unit (the magnetic coil) triggers the wearable. The accelerometer is then activated, recording the acceleration values every 3 seconds over a 12-second interval. If a separate motion trigger is fired during this 12-second period, the activation time is extended again by 12 seconds. The wearable logs all the collected accelerometer data locally using a FRAM storage. At the end of the study episode, all the recorded samples are transmitted back to the AP and the voltage at the capacitor is measured. A comparison of the energy stored in the super-capacitor before and after the experiment helps to determine if the overall harvested energy is

sufficient (or not) to support the sensing, local storage and 'ping' packet transmission tasks. We compute the stored energy in a capacitor using the equation:

$$U = \frac{1}{2}CV^2; \quad P = \frac{\Delta U}{T}; \tag{3.2}$$

where C is the capacitance (=0.47F), V is the voltage in Volt, U is the stored energy in Joule, P is the 'average' power in Watt, and T is the observation duration (30 mins). In other words, P represents the power differential (averaged over 30 minutes) between the harvested and expended power.

As illustrated in Figure 3.21, the AP is placed behind the table, the 8 antennas are raised up to 0.9m and point down to the middle of the table (45°). Figure 3.23 plots the average differential power (the net change in super-capacitor energy, divided by 30 minutes) of each user. We observed differences in the activities performed by the 4 users: one user read paper-based text; one worked on his aluminum-bodied laptop (primarily reading content), while two used their smartphones. As expected, the device on two users located at the corners closest to the AP harvest the highest power and end up with the largest positive residual power.

However, the AoA estimation of the second user (see Figure 3.24) is sometimes quite inaccurate (maximum error = 32°), in contrast to the other users whose error is usually $\leq 5°$. We suspect that the RF reflectivity of the laptop's aluminum body may be a contributory factor. Figure 3.25 shows the total dura-

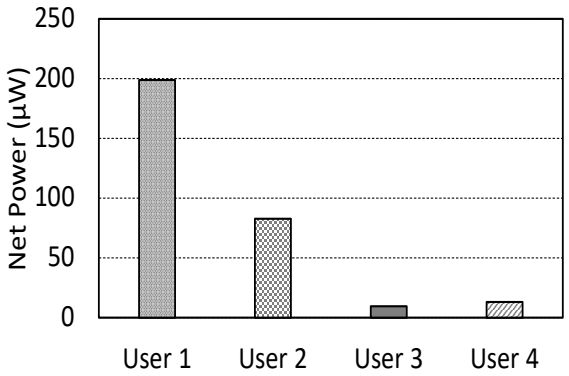


Figure 3.23: Net energy for 4 users (distance = {1.3, 1.4, 2.2, 2}meters)

tion for which each user's accelerometer data is recorded. We also notice that the user reading documents on a laptop (user 2) triggers the system less often than the user reading paper-based document (user 1), likely due to the greater hand motion involved in turning pages on a physical document. While the 3rd user (User 3) simply performs browsing on his smartphone, User 4 is much more active, resulting in the capture of over 4 minutes of his hand movement. Note that even though these users are located farthest from the AP (2.2m and 2m respectively), the wearable device still ends up being net-power positive.

Overall, our results demonstrate that the WiWear's *batteryless wearable can indeed continually monitor the key hand movements of users, as the harvested energy is greater than the expenditure in all 4 cases.* This energy-positive operation occurs even though our current linear antenna is known to have very low gain near its poles; the use of better antenna designs should further enhance the energy harvested.

3.6 Discussion

While our results attest to the promise of *WiWear*, there are, however, several open issues to explore further.

Non-Line-Of-Sight (NLOS) operation: In the case of NLOS, the simple beam-forming technique might not work. The system might need to scan for the optimal phase of antennas (similar to EnergyBall [51]), but continuously

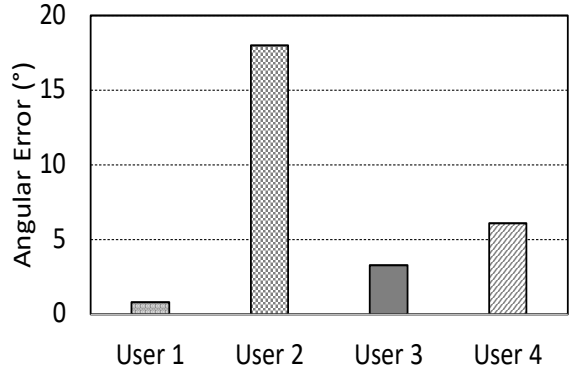


Figure 3.24: AoA error (4 users)

transmitting feedback to the AP drains the energy of the device. This method is promising to enable NLOS operation of static devices as one-time calibration is usually sufficient (until the environment changes). For a dynamic device (a wearable), the device is likely to move between LOS and NLOS, and additional experiments are thus needed to evaluate the performance loss due to NLOS operation, across a variety of environments and device usage contexts.

Alternative low-power channel access: The CSMA (Carrier Sense Multiple Access) mechanism requires an RF transceiver to stay active and listen to the channel, so it is an extremely power hungry operation in any RF transceiver. CSMA in a low-power RF transceiver is almost unusable in a WiFi compatible system as its response time is much higher than the DIFS (distributed inter-frame space) of WiFi standard. Without a proper channel access protocol, a device cannot transmit packets back to an AP reliably and efficiently. Ultra low-power wake-up receiver (WuRx), which has initially been introduced in wireless sensor networks, potentially provides an alternative low-power channel access mechanism. WuRx operates at lower frequency, and consumes much less power (less than $5\mu A$) compared with devices working at WiFi band. WuRx does not detect "idle" channel, but detects specific signal patterns from AP which indicates that it is safe to transmit data back to the AP. An AP may utilize this additional channel and normal WiFi channel to coordinate multiple WiFi and WiWear devices.

Multi-AP Operation: In a practical campus or factory environment, multi-

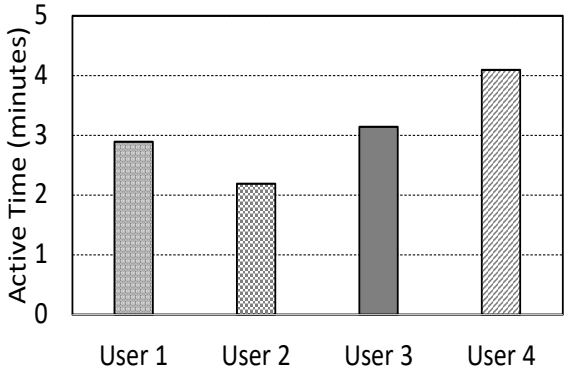


Figure 3.25: Active accelerometer sensing period

ple APs are likely to 'cover' a specific location (e.g., in our campus, the number of APs overheard at a typical location is 5-6). This opens up additional possibilities. Clearly, as illustrated in EnergyBall [51], transmission of suitably beamformed transmissions on a common channel, along with careful phase control, can significantly increase the harvested energy. Alternately, each AP can transmit its 'power packets' on its own independent channel, thereby eliminating the difficult task of phase synchronization. However, the RF harvester module on the wearable must be enhanced ([119]) to (a) allow the harvester to simultaneously support multiple resonant AP frequencies, and (b) implement dynamic impedance matching (e.g., [52]).

Power vs. Throughput Tradeoffs: Our early results (Section 3.4.D) show that there is a tradeoff between the two objectives of data transfer and RF charging that remains to be explored. Additional mechanisms may be used to optimize this tradeoff: e.g., adjusting the schedule & duty cycle of power packet transmissions (e.g., by using multiple virtual queues [105]) to avoid unacceptable loss or latency of data packets or transmitting data packets at higher power (for enhanced energy harvesting).

Additional & Improved Energy Harvesting: The current *WiWear* prototype uses a basic whip antenna for energy harvesting (gain=2.1 dBi), whose performance degrades for large values of either the azimuthal or elevation angles. It is very likely that alternative antenna designs (e.g., a metallic strip-based "patch antenna) can increase the harvested energy significantly (see [44]). Moreover, wearables may combine WiFi energy harvesting with other alternative harvesting techniques, such as ambient light, for significantly improved performance. Also, the magnetic trigger may be replaced with a kinetic energy harvester (such as the ones used in mechanical watches) that also harvests additional energy.

Other Application Domains & Paradigms: Our investigations focused on a single AP, with a single user in an office-like setting. While our experiments were limited to an office setting for reasons of practical feasibility, we

believe that our experimental setup is conceptually similar to that of our target use case—one where a user attaches multiple such devices to her limbs and engages in a real-time interactive rehab or game-training application inside a confined room. Additional research is needed to apply the core *WiWear* concept to other scenarios, such as (a) capturing key locomotion and gesture-related behaviors (e.g., fall detection) of elderly inhabitants in smart homes; (b) operating static sensors, deployed in industrial sites and warehouses.

3.7 Reflections and Lessons Learned

Throughout the development of *WiWear* system, I have gained several valuable lessons and insights on the challenges of working with embedded devices and components. I list a few of my key learning points, such that future researchers may avoid some of the pitfalls that I encountered.

Your touch might cost a lot of time: We might think that touching a circuit board when it is completely unplugged is safe. However, when working with ultra low-power devices, usually there are devices whose input resistor is extremely large (several $M\Omega$). Our finger has sweat which may be even more conductive than the input resistor, and the left-over sweat functions as a parallel resistor which changes the input of a device significantly. In the worst case, it may kill the device once it is powered again. Likewise, even your exhalation can also kill a device. The problem is that we may look over these effects and spend a lot of time investigating other causes. Covering the sensitive part of a device using polyimide film can avoid this problem.

Floating input might cause big trouble: When working with embedded system, usually we have to work with floating input (not connected to either ground or voltage source) of a pin of an IC. There are cases, leaving the pin floating will drain a lot of energy (if it is a control port) which is crucial for low-power system. To avoid floating value, one may use a resister to connect it

to source voltage or to ground line. But if this connection is not done carefully, it may eat up the energy of the system silently. For example, a $10K\Omega$ resistor is connected from an input pin of component A to 3.3V source to maintain a logic 1 level in normal condition. Another component B also connects to this pin as a trigger line. The output of component B is usually 0, only becomes 1 to trigger the component A. So this connection silently consumes 1mW ($3.3^2/10000$). So always maintaining a specific level 0 or 1 for input pins, and avoiding the case of an implicit source-to-ground connection will protect the device from energy leakage.

Chapter 4

Fine-grained Real-time Motion Sensing

Inertial sensing on wrist-worn devices, such as smartwatches, has recently been used to infer a variety of gesture-driven lifestyle activities, such as smoking [94] and eating [112, 121, 25]. These approaches typically focus on the problem of gesture recognition, i.e., using features defined over the inertial sensor data to identify specific gestures. Separately, researchers have investigated the use of such possibly-noisy inertial sensor data to track the hand's 3-D location or movement trajectory, for applications such as pointing based interaction [98] and handwriting recognition [24].

In this chapter, I investigate the possibility of using such wrist-based sensing (e.g., via a smartwatch) to enable a novel class of pervasive, real-time, gesture-driven interactive applications, such as immersive virtual reality (VR) games. For these applications, the objective is to accurately track the movement of the human hand, but with latencies low enough to preserve the interactivity of the application or the game. As an exemplar of such a gesture-centric game, consider a VR-based Virtual Table Tennis (**VTT**) application (corresponding to the use case 2 in Section 1.2.1), as illustrated in Figure 4.1. Here, the user is either competing or training against an opponent. While the user sees the table ten-

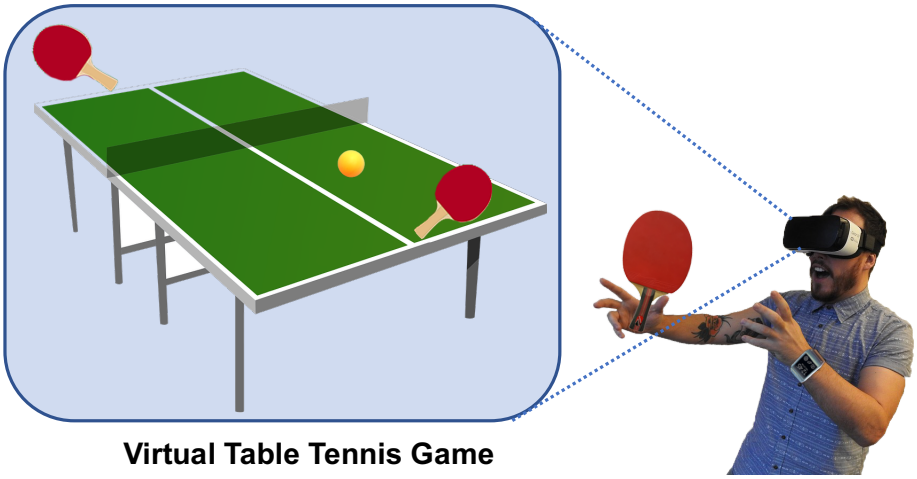


Figure 4.1: The Virtual Table Tennis (VTT) application. A user makes real-world TT gestures while wearing a smartwatch, with the gestures being integrated into the virtual world displayed on the wearable VR device.

nis board, the ball and the opponent in the VR display, she uses the real-world physical movement of her hand (wearing a smartwatch) to hit the ball. The hand motions are tracked and integrated into the virtual world, and appropriately projected in the VR display. Accurate tracking of the hand movement is needed to faithfully replicate real-world mechanics—for example, the time instant when the racket hits the ball in VR should closely reflect the time that the user's hand would have struck the ball in the real world.

Achieving these goals requires overcoming several challenges. The key characteristic of the target class of immersive applications considered is that they require us to both (a) classify individual gestures and (b) to concurrently track the hand's movement trajectory, in real time. To provide the user with specific training on the type of stroke played, the system needs to correctly identify the stroke; moreover, to provide accurate visualization of *when* and *where* the user hits the ball, the system needs to track the hand's trajectory as well. A closer analysis of the representative interactive application reveals the following unique challenges:

- *Calculate the Trajectory Fast and Accurately:* Table Tennis is a fast-paced game, with professional grade players often exchanging $\approx$ 120 strokes

each per minute (see [142]) during a rally. To provide a truly interactive feel, the game must not exhibit *lag*—i.e., the game must detect the instant of contact between a player's racket and the ball instantly, so that the virtual reality game can proceed apace. This means that we must not only compute the hand's trajectory fast, but also precisely (as the calculated point of intersection between the racket and the ball will affect the calculated time instant of contact as well).

- *Recognize the Gesture Before It is Completed:* Most stroke-based games (e.g., tennis, table tennis and badminton) involve a significant "follow-through"—i.e., the actual game gesture involves significant movement of the hand both before and after the act of striking of the ball. Accordingly, we cannot afford to delay the execution of the recognition step till the end of the gesture, as this would seriously impact the interactive feel of the game.

The above requirements thus require an entirely new class of gesture recognition techniques, that can recognize gestures fast (even before the gesture is complete) and simultaneously track the hand's trajectory accurately, in real-time. In this chapter, I develop an integrated smartwatch-based gesture recognition framework that tackles these two objectives: (a) gesture recognition and (b) hand tracking concurrently, and with low latency.

4.1 Representative Application & Requirements

The requirements for an enhanced gesture recognition cum trajectory tracking solution come from our vision for a new class of multi-device immersive applications. In these applications, the user relies on multiple mobile & wearable devices, whose input and output interfaces are combined to offer an integrated, multi-modal experience. In particular, we consider the class of VR or AR (augmented reality) applications, where the user experiences a virtual/augmented

world on her smartglasses, with other wearable devices providing fine-grained tracking of the user's gestural activities.

In the representative VTT application (illustrated in Figure 4.1), the wrist-worn smartwatch (or smartwatches) are used to monitor the fine-grained movement of the arm. The inertial sensors on the smartwatch can then be combined with the inertial sensing data from the head-mounted smartglasses, to obtain the movement trajectory of the arm *relative* to the user's body orientation, and this movement trajectory can then be embedded inside the virtual world displayed on the smartglasses (e.g., a point-of-view representation of the user's avatar). The VTT application can then be used to provide a realistic emulation of two (or four) players playing Table Tennis without being in physical proximity, with the players sharing a common view of the court and having to play actual strokes in the real world. Furthermore, to provide more realistic feedback of the experience of a real Table Tennis game, we can envision that the wrist-worn device provides some form of tactile feedback (e.g., by vibrating the wearable device) whenever the system detects that the user's racket has hit the ball. To provide a high-quality user experience, the tracking of each user's strokeplay should be of *low-latency* (to ensure that the virtual world rendering and the tactile feedback do not perceptibly lag the physical world gestural activities) and *accurate* (to ensure that the outcome of the strokes reflect the physical world with high fidelity).

While the investigations in this chapter are confined to this representative VTT application, the general requirements for low-latency gesture recognition and trajectory tracking apply to a much broader class of applications. For example, consider other VR-based applications such as a Dancing Tutor (where a user's dance-related moves are tracked and projected into the virtual world) or a Racing Emulator (where a driver is assumed to steer a racing car using a virtual steering wheel, with the wearable sensors providing real-time vibratory feedback about the road surface conditions). Another application could be real-time

interactive feedback for gesture-based rehab training, a scenario detailed as use case 1 in Section 1.2.1. All of these applications exhibit the previously described characteristics of (i) a critical interaction occurring not at the end, but at an intermediate point, of a gesture, and (ii) the need to accurately track a user's arm movement, in real time, but only for a finite set of application-relevant gestures.

4.1.1 Perceiving Latency and its Effects on Usability

Before proceeding further, it is useful to understand and quantify the desired performance characteristics of the representative VTT application. In particular, I am interested in examining the latencies generated from *gesture recognition* and *hand trajectory tracking*, and understanding the point at which the latencies becomes noticeable to the user, thus impacting the user experience.

Jota et al. [62] examined the effects of latency on direct-touch pointing tasks. They found that no participant could notice latency below 20ms, with most participants (85%) not being able to differentiate between 1-40ms of latency. I conducted experiments with 12 participants to examine the *noticeability of latency* within the context of VTT. Participants were asked to hit a suspended table tennis ball with a table tennis bat 30 times. On detecting the hit event (this was measured via visual tracking with a high-speed camera (100 fps), attached to a powerful desktop that achieves near-zero frame processing latency), an audio alert was played after a randomly generated delay ranging from 0 to 500ms. I take the time difference between the audio alert and the hit event to be the *lag/latency,* and asked participants to indicate if they noticed the delay.

Figure 4.2 plots the cumulative distribution of the probability of user perception (i..e, the fraction of instances where the user indicated a perceptible delay) as a function of this lag/latency, for the experiment described above.The results are similar to Jota et al., with no user being able to perceive latencies below 19ms, and only 5% of the noticed latencies being between 19-40ms. We observe that 80% of users are completely oblivious to lags less than or equal to

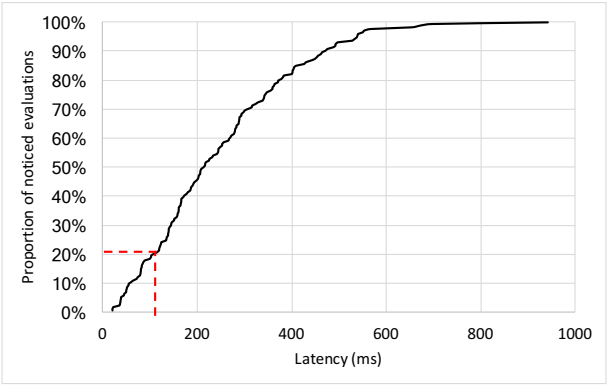


Figure 4.2: Cumulative distribution of noticed latencies.

100 msec, and thus set ourselves a benchmark of developing a tracking system that can identify the specific stroke within 100 msec of the actual hit event.

4.2 System overview

To realize the vision of a practical gestural interface which can both recognize gestures early and track user's wrist accurately, I propose a system architecture (Figure 4.3) where a *Continuous Progressive Gesture Recogniser* cooperates with a *Gesture Based Trajectory Tracker* to enable early detection of gestures and infer the position of user's wrist during a gesture.

This system uses only sensors on a smartwatch without any environment/infrastructure installment. The *Data Collector* collects Accelerometer, Gyroscope, Magnetometer and additional derivative sensors (which are computed from the three physical sensors)–i.e., the Rotation Vector, Game Rotation Vector, Linear Acceleration and Gravity. The Rotation Vector, Linear Acceleration, and Gravity values are conveyed to the recognizer. The Rotation Vector is conveyed to the Trajectory Tracker. In general, the system includes the following two key components:

- *Continuous Progressive Gesture Recogniser* (using a combination of HMM+ Classifier): Firstly, a *Feature Extraction* block extracts the Arm motion features including the velocity in vertical and horizontal direction

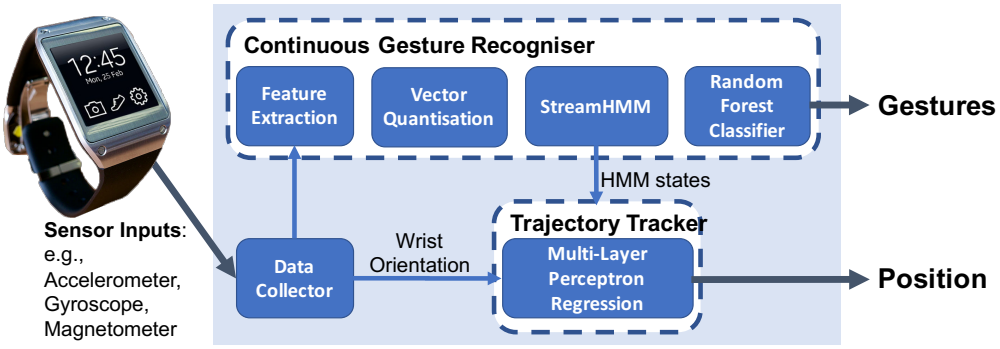


Figure 4.3: System Overview.

and the angular orientation of the lower arm. These features are then concatenated as a feature vector and are quantized into discrete values by the *Vector Quantization* block. These discrete values are fed into a modified HMM model (*StreamHMM*) to compute the probability of gestures states. Finally, a classifier (Random Forest) is applied to enable the early detection of gestures.

- *Gesture-Based Trajectory Tracker:* The orientation (values of the X-Axis of the smartwatch) is extracted from the sensors by *Data Collector*, and is used as one feature of the *Trajectory Tracker*. It also uses the probabilities of the HMM states (of the Recogniser) to improve the tracking accuracy. A regression model uses the gesture states and instant wrist orientation as features to accurately estimate the position of the wrist.

Our early gesture recognition and hand tracking technique is proposed based

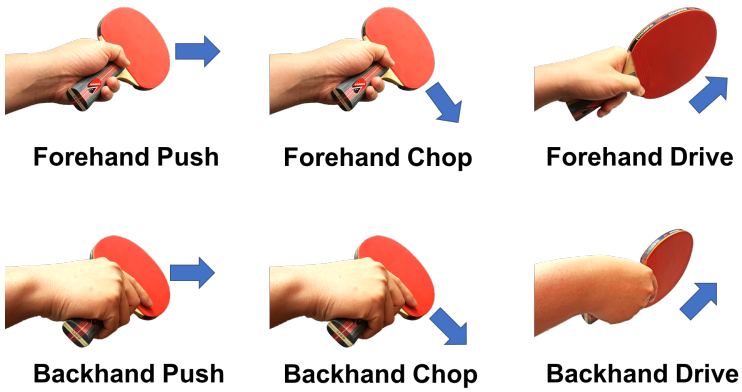


Figure 4.4: The six table tennis strokes used in the study. The arrows show the direction of motion as applied to a table tennis ball coming from the right.

on the following assumptions:

- *Hand tracking for only legit gestures:* As our hand tracking method is based on the gesture recognition, the hand position estimation during other movements, which are not legit gestures, are invalid. Applications, which require hand tracking of arbitrary movements (e.g, free drawing) might not benefit from our technique.

- *No consecutive gestures:* Our gesture recognition model uses a *universal-state* to automatically re-calibrate state probabilities. It implies that each gesture should return to non-gesture state before another gesture may start. This is not expected to be a major problem for most sports-based or therapy-related gesture movements, where the user usually performs a set of discrete gestures (e.g., table tennis strokes, swings of a golf club, hand stretches during therapy), punctuated by intervals of non-gestural activity.

- *Minimum gesture length of 110 ms (11 samples):* As each gesture is modeled as a 10-state HMM sequence with a return-connection from the last state, each gesture is expected to have a length of at least 11 sample (110 ms with 100Hz signal) to reach the last state and return to the *universal-state*. This minimum gesture length is much shorter than the average gesture duration of common stroke-based games such as table tennis (510ms) [138, 37] or golf (836ms) [116], and even shorter than the duration of forward-swing phase (moving hand forward until the point of contact) which is 170ms and 387ms with table tennis and golf strokes respectively.

4.3 Dataset

I conduct two experiments to collect sensor and ground truth positional data, while participants perform the six basic table tennis strokes (see Figure 4.4). These studies are reviewed and approved by our Institutional Review Board

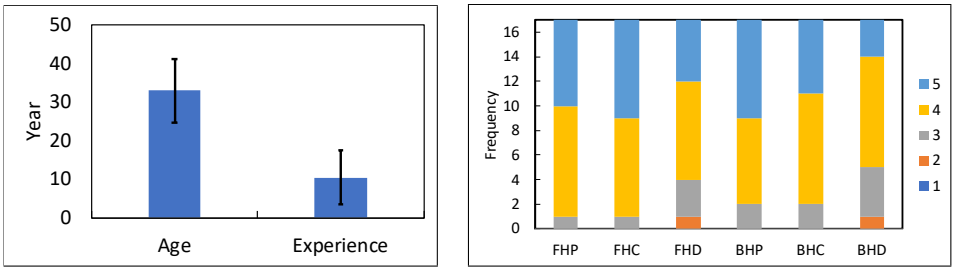
(IRB-16-127-A131-M2(817)). The sensor data is recorded using a smartwatch on a participant's wrist, and contain 3 hardware sensors: *Accelerometer*, *Magnetometer*, *Gyroscope*, and 4 derived sensors: *Linear acceleration*, *Gravity*, *Rotation vector*, and *Game rotation vector*. The ground truth positional data contain RGB video and the position corresponding to each pixel in the video. We use a ZED camera [117] mounted on the ceiling of the experiment room to record the positional ground truth data. To later extract the position of the wrist and head, I provided a yellow band and a pink band and asked participants to wear the yellow band on their head and the pink band on their wrist (wrapped around the smartwatch). *Note that the head/wrist bands, as well as the camera, are merely used to collect ground truth data for validation, and do not form part of our proposed wearable-based recognition technology.* I use a RoboPong 1050[1] table tennis robot to automatically serve balls to our participants. The robot was placed at the other end of the table (see Fig. 4.6).

In the first experiment, I recruited 10 participants, without any criteria on the experience or skill level of the participants. All participants were male university students in the age group of 21-30 years and had a working understanding of how to play table tennis. Nine of them did not play on a regular basis, and one used to play on average once a month. I used a *Samsung Gear Live* smartwatch to record sensor data at a sampling rate of 100Hz. At the beginning of the experiment, participants watched 6 training videos of the 6 gestures on a table tennis training channel [99]. Participants were given a short period of time ($\sim$5 minutes) to get familiar with the robot and gestures until they feel comfortable to play. For the study task, participants were asked to execute the strokes with balls served by the robot to achieve **24** successful returns per stroke. A stroke is deemed successful if the participant is able to return the ball to either the left or right halves of the opposite side based on our prompting from a balanced, randomized schedule (e.g., Left, Right, Left, Left, Right, Right). If a partici-

[1] https://www.newgy.com/p-279-robo-pong-1050-plus.aspx

pant fails to return the ball to the correct half, he is asked to repeat that stroke until he is successful. The focus on successful stroke completion is intended to encourage participants to perform the gestures conscientiously, and to provide a sufficient number of usable gesture instances. As such, while a stroke may be deemed unsuccessful, the attempt can still be considered as a data point for that stroke. I examined the captured data, post-session, to remove invalid stroke instances (i.e., warm-up strokes). The table tennis robot was configured to serve balls at consistent spin, and to serve balls to two positions for each stroke, to increase the variation of gestures.

Results from the first experiment achieved a high average gesture recognition accuracy, but also exhibited higher variance. Given that most participants had insufficient playing experience, it was unclear if the gestures performed were representative, and if the results obtained were affected by the likely variability in stroke-making among the participants. Accordingly, I conducted a second experiment, where I collected sensor data from 17 *experienced* players, recruited from an online table tennis group, where I explicitly indicated the need for a minimum of 3 years of playing experience. The group had only one left-handed player and one penhold-style player, whom we've excluded from our studies due to the low sample size. Figure 4.5a plots the age and experience of the remaining 15 participants, who were 23–56 years old, with an average age of 32.8. The least experienced players have played table tennis for 3 years, while the most experienced players have 25 years of table tennis experience. Except for one player who plays table tennis daily, the others play table tennis weekly. Prior to playing, users were also asked to perform a self-evaluation of their proficiency in the 6 gestures (illustrated in Figure 4.5b), using a Likert scale ranging from 1-5: (1) corresponds to *I don't know how to play that stroke*, while (5) corresponds to *I am expert at it*. For these set of experiments, I used a more-modern *LG Urbane W150* smartwatch to record sensor data with the same sampling rate as in the first dataset. Participants were not asked to watch



(a) Participants' age and experience. (b) Self evaluation of proficiency.

Figure 4.5: (a): Average age and experience of participants. The least experienced participant has 3 years of experience. (b): Proficiency of participants in the 6 gestures based on self evaluation. (1) means "I cannot perform the stroke", (5) means "I am expert in this stroke".

the video again because they had seen it during the online registration form. However, they were allowed to practice some shots to become familiar with the robot. Each participant in this study was asked to finish **30** successful instances of each of the 6 strokes.

4.4 Early Gesture Recognition

In this section, I tackle the problem of recognizing gestures *accurately* and *early*–i.e., before the entire gesture has been completed. In Section 4.4.1, I discuss why the conventional approach for continuous gesture recognition, involving the cascaded steps of segmentation and classification, is insufficient, Accordingly, I propose and evaluate an alternative approach with two novel characteristics: (a) a unified HMM model, with a novel *universal* state that allows a gesture to be recognized in a streaming fashion, and (b) an explicit additional classifier, which uses features defined over the evolving HMM states, to recognize gestures early and robustly.

4.4.1 Inadequacy of Explicit Segmentation

Conventional gesture recognition techniques usually combine two stages: *segmentation* and *classification*. *Segmentation* first partitions the incoming sensor data into consecutive segments; each individual segment is then classi-

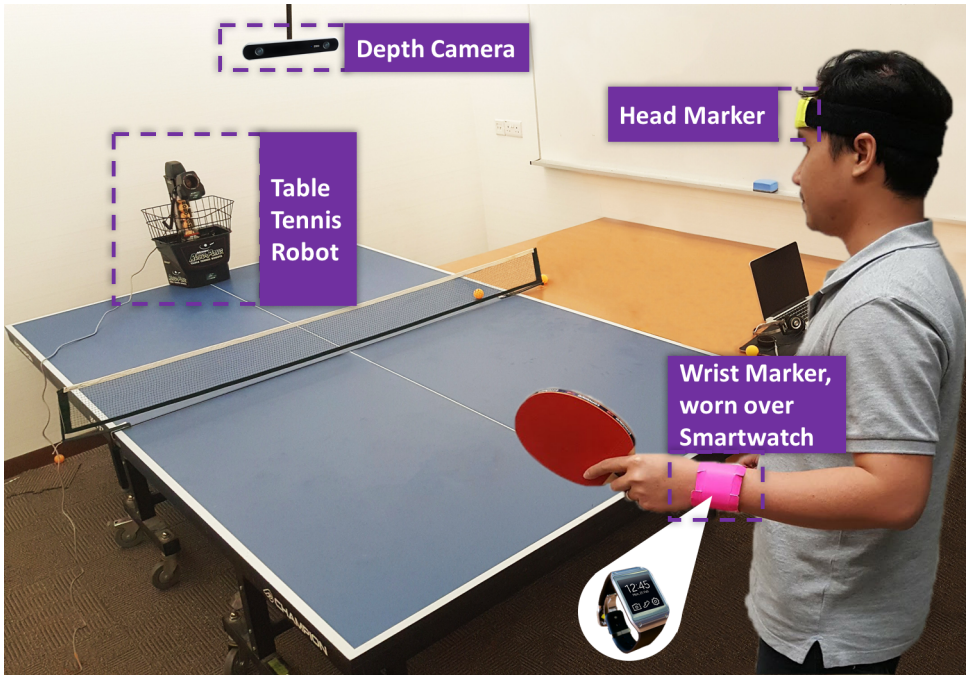


Figure 4.6: Experiment Setup. The participant plays with a table tennis robot, while a high speed depth camera captures ground truth position data. A smartwatch worn on the hand provides sensor data on strokes performed.

fied to assign it a particular gesture label. Gesture classification is typically performed using techniques such as Dynamic Time Warping (DTW) [80] or HMMs [109, 96].The overall accuracy of gesture recognition thus depends on both the segmentation and classification steps.

The basic segmentation technique is based on thresholds, with the gesture deemed to have started when some sensor-derived value exceeds a threshold and to have ended when the value falls below another threshold. Typical examples include setting the threshold on the hand acceleration magnitude (derived from accelerometer readings) and/or rotation values (derived from gyroscope readings). More sophisticated techniques can set these thresholds adaptively; for example, the authors in [96] propose a method of adaptively adjusting the acceleration threshold based on the observed false-positive (FP) and false-negative (FN) rates of the accelerometer segmentation compared with gyroscope segmentation.

This pipelined model of "segmentation followed by classification" suffers from three drawbacks:

1. *Longer Latency:* By definition, a segment can be identified from a sensor stream only after the last sample for the segment has been collected—i.e. after the associated gesture has been completed [63]. Moreover, segmentation algorithms (e.g.,E-Gesture [96]) often impose additional latency by requiring an additional observation period (beyond the end of the current gesture) to guard against the problem of 'gesture splitting', where a more complex gesture is inadvertently split into several segments. This approach is thus problematic for interactive gesture-based applications, where users expect the system to essentially react "instantaneously" (in real-time). More specifically, for our VTT application, we notice that a player typically hits the ball halfway (40%) into the gesture.

2. *Poor Segmentation Accuracy Under Large Differences in Gesture Dynamics:* Threshold-based approaches typically work well when the different gestures have similar dynamics, such as similar overall duration or initial hand-force intensity. However, finding an appropriate threshold value is challenging, when the dynamics of gestures vary considerably (as is the case for table tennis, where certain strokes involve rapid and extended hand movement, while other strokes, for e.g., Backhand Push, involve significantly lesser movements of the hand). To illustrate this point, I evaluated the accuracy of a single threshold-based segmentation technique (similar to E-Gesture, the mechanism outlined in [96]) on table tennis gestures. Figure 4.7a shows the accuracy of the table tennis gesture data set as a function of the acceleration threshold.

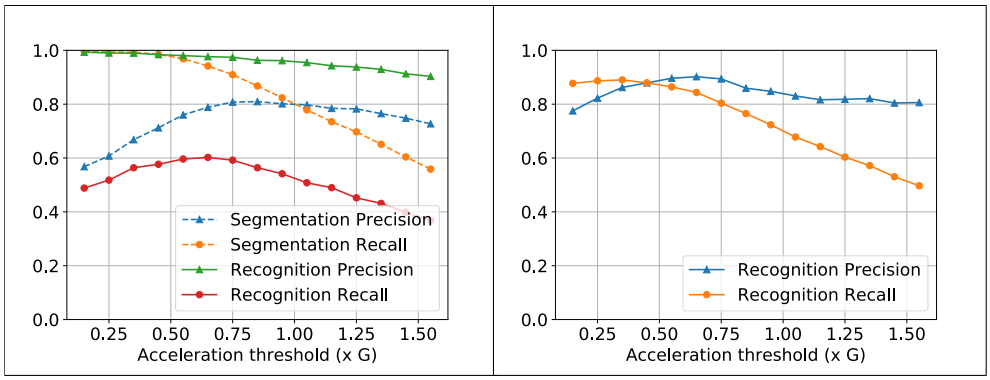
   The figure plots both the segmentation recall/precision (i.e., determining how many segments were identified compared to the ground truth), as well as the classifier recall/precision (what fraction of the segment duration was truly a part of the associated gesture?). Here precision is defined as: $\frac{\sum TruePositive}{\sum TruePositive + \sum FalsePositive}$, while recall is defined as: $\frac{\sum TruePositive}{\sum TruePositive + \sum FalseNegative}$. We notice that with low threshold, the seg-

mentation step rarely misses gestures, but the segments contain redundant data which will lower the classification recall. With high threshold value, the segmentation step misses gesture more frequently, and thus also results in low classification recall. In general, using E-Gesture, the system gets the highest recall of about 0.6 at a threshold of 0.65G. The precision is quite high and does not change considerably.

3. *Inaccurate Segmentation, Leading to Poor Classification:* Even if the segmentation process can identify the occurrence of a gesture, it can misalign the start or end samples of the gesture–i.e., it can either truncate part of the entire gesture (in this case, the identified segment of samples is smaller than the true gesture duration) or it can falsely append spurious samples to a gesture (now, the segment duration is longer than the true gesture duration). This misalignment can reduce the accuracy of the downstream classification step. This can be observed by noting the classification recall and accuracy values presented in Figure 4.7a. We see that the classifier precision decreases slightly but still remains higher than 90% as the threshold is increased; however, the recall is fairly low (mostly below 60%) indicating that segmentation typically misses (truncates) a significant portion of individual gestures. A careful inspection of our dataset showed that a fixed-threshold segmenter tended to append spurious samples to gestures that involved more vigorous hand movement (such as Forehand Drive and Backhand Drive). In contrast, the segment was often truncated for less vigorous gestures (such as Forehand Push and Backhand Push).

Improved Baseline for Segmentation-based Classifier:

Our initial results showed that the gesture classification accuracy was low, often because the segmentation step included spurious non-gesture accelerometer readings before and after the true gestural segment. More specifically, the HMM models in [96] output the final gesture only after the sensor data of the entire



(a) e-Gesture Performance (b) Enhanced e-Gesture Performance

Figure 4.7: (a): Applying E-Gesture technique [96] into our Table Tennis dataset. Threshold values range from 0.15G to 1.55G. The recall reach highest value of 0.6 at a threshold of 0.65G; it means only 60% true gestures are recognized. The precision is quite high, more than 0.9 accordingly, and slightly decreases when the threshold increases. (b): E-Gesture based enhanced classifier that uses the highest confidence value that an HMM's state sequence achieves at *any intermediate* point of the segment. Results in significantly higher recall, at the expense of reduction in precision.

segment has been processed. Due to the spurious trailing non-gestural sensor data, the confidence level of the state sequence of each HMM classifier would fall below the confidence threshold.

To remedy this issue, I developed an alternative personalized model, where the confidence levels of the hidden states were tracked throughout the evolution of the identified gestural segment. More specifically, a per-HMM counter (each HMM corresponding to one of the 6 distinct gestures) was used to keep track of the *maximum confidence* value associated with the state sequence of each HMM, at *any* point during the segmented gesture. In the other words, this model computes the confidence value (probability) of each HMM on the current sub-segment until the end of the segment is detected. The segment was subsequently classified to be the gesture that matched the longest sub-segment with the highest confidence value.

Figure 4.7b plots the resulting precision and recall for this improved classifier. We observe a better balance of precision and recall. The recall value improves because this approach can effectively ignore the spurious motion artefacts that follow the true gesture. However, the precision drops (the false-

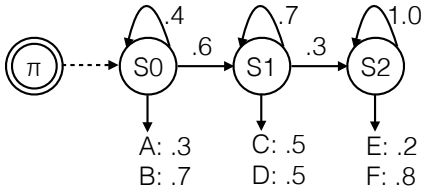


Figure 4.8: Left_to_right HMM is useful to infer the states of a finite segment of data. {A-F} indicate the various output (i.e., observable) values–i.e., a vector of sensor values/attributes that are observed in each of the hidden states. The associated number denotes the emission probability.The initial state π serves as a starting point of the forward and Viterbi algorithm.

positive rate increases) as this approach also results in several non-gesture segments being classified as a legitimate gesture, especially if parts of the segment had motion that bore a resemblance to one of the gestures. Based on empirical analysis, we found that a threshold of 0.45G served as an effective choice: yielding a recall value of 87.9% and a precision value of 87.9%.

4.4.2 Segmentation-less Gesture Detection on Stream Data

I now describe our novel approach for gesture detection that dispenses with the aforementioned segmentation step, and instead continually tries to identify gestures that occur within an incoming stream of sensor data.

Our approach is based on appropriate modifications to the basic Hidden Markov Model (HMM) based recognizers. HMMs have been widely used for gesture recognition (e.g., [109, 96]). The Left-To-Right HMM (See Figure 4.8) has been popularly used, as its structure matches well with the sequential evolution of gestures. In the traditional technique (which required a preceding segmentation step), each gesture is modeled as an HMM. Whenever a segment is identified (e.g., consisting of the inertial sensor samples $\{X_0X_1..X_t\}$, the probability of the segment with respect to each HMM is computed using Viterbi

algorithm (4.1) as follows [2]:

$$P_s^0 = \pi_s * B[s][X^0]; P_s^t = \max_i P_i^{t-1} * A[i][s] * B[i][X^t] \tag{4.1}$$

This classical HMM approach requires an explicit starting state (corresponding to the start of the gesture) and operates on a finite length segment. We thus need to modify the HMM model to account for an infinite stream, with no explicitly delineated gesture segments.

In this work, we modify an HMM-based approach, as an example of a probabilistic state machine. Note that there are more sophisticated recent techniques, such as *Long Short Term Memory* (LSTM) neural networks, that have been proposed for processing sequential sensor data. In this work, our primary focus is on developing an overall framework for low-latency gesture detection and 3-D tracking; hence, we select the relatively low-complexity HMM as our candidate recognizer, as it can be executed on a smartwatch with very low processing overhead.

Our approach is based on the integration of a new *Universal State* into an existing HMM model (See Figure 4.9). The goal of the *Universal State* is to allow a "regular-expression like" matching of a gesture signature (an approach previously investigated in [118]), within a continuous sequence of observations with arbitrarily long non-gesture periods. In effect, the *Universal State* serves as a wildcard character (*), capturing the arbitrarily long sequence of non-gestural observables. The *Universal state* generates observations with the same probabilities of observations in the entire dataset–in the simplest case, this can be a uniform distribution, while it can also reflect the observed probability of a gesture occurring over the entire observation period. The *Universal state* also has a self-transition loop, implying that the HMM can stay at *Universal state* for an indefinite time; this corresponds to the arbitrary interval between two consecu-

[2] A[s][ns]: Probability of a transition from state *s* to state *ns*. B[s][O]: Probability of state *s* generating observation *O*

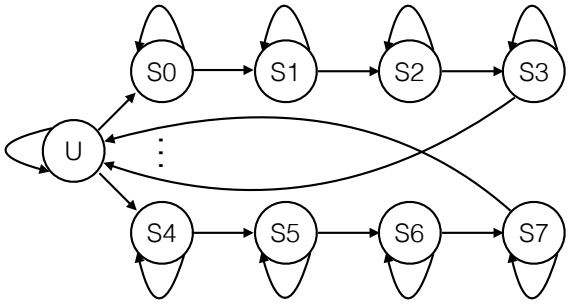


Figure 4.9: Left_to_right HMM is useful to infer the states of a finite segment of data. {A-F} indicate the various output (i.e., observable) values–i.e., a vector of sensor values/attributes that are observed in each of the hidden states. The associated number denotes the emission probability.The initial state π serves as a starting point of the forward and Viterbi algorithm.

tive gestures. To reduce the chance that the observations are assigned to the final state of a gesture model for a long period, a return connection is added from the final state of each gesture model to the *Universal State*. Finally, whenever the system receives an observation (a vector of features), it updates the probabilities of states using equation 4.2 and 4.3. Initially, the probability of *Universal State* is 1.0; S_i^t is state i at time t, O^t is observation at time t.

$$P(S_i^{t+1}|O^{t+1}) = \frac{P(O^{t+1}|S_i^{t+1}) \times P(S_i^{t+1})}{P(O^{t+1})} \tag{4.2}$$

$$P(S_i^{t+1}|O^{t+1}) = \frac{B[i][O^{t+1}] \times \sum_1^n P(S_j^t) \times A[j][i]}{P(O^{t+1})} \tag{4.3}$$

Features used in the VTT Application

To build a working implementation of our VTT application, we need to define the sensor-driven set of *features* (i.e., the observation vector O in Equation 4.3) that we use to compute state transition probabilities. Prior work on HMMs for gesture recognition has defined O using relatively simple features over the raw data, such as discretization, quantization[80, 109], or taking differentials or integrals of accelerometer and gyroscope data[96]. In our work, we utilize a small set of more-complex *relative orientation* features, computed from the underlying accelerometer and gyroscope data as described in Table 4.1. These

Feature	Description
Horizontal velocity	Arm's angular velocity in horizontal plane
Vertical velocity	Arm's angular velocity in vertical plane
Wrist vertical angle	Wrist's rotation angle compared with vertical line

Table 4.1: Features used to define the output vector (O) for HMMs

features are then quantized to 256 discrete levels, using a K-Means module with 256 centroids.

4.4.3 Early detection of gestures

I now describe the cascaded HMM-cum-classification technique that forms the core novel component of our early gesture detection algorithm. The Universal-state augmented unified HMM model allows us to detect the occurrence of candidate gestures in a streaming fashion, without requiring a separate and explicit segmentation step. However, in its base implementation, the unified HMM model (Figure 4.10) still identifies a gesture only when it is complete–i.e., only when the corresponding Left-to-Right model has its end-state exceed a specified likelihood threshold. Additional modifications to the gesture detection logic are needed to support our second objective: *of detecting a gesture early*, even before the entire gesture has been completed.

The simplest way of inference using this model is to define a threshold of probability. The system will select the model whose probability is the highest among gesture models. If this probability is higher than a threshold, the system outputs the corresponding gesture, otherwise, it will output *null-gesture*. However, the specification of a single explicit probability threshold value does not work across different gestures that have very different kinetic vectors.

Accordingly, we do not use an explicit threshold to perform early gesture detection. Instead, we develop a novel technique, where the *probability values of the hidden states of the HMM are used as features* by a supervised classifier to generate the output gesture. More specifically, for *each* individual Left-to-Right HMM (corresponding to a single gesture), after every incoming

Type	Feature	Description
Model	Normalized Expected State	$\frac{\sum_{i \in S} p_i * s_i)}{\sum_{i \in S} p_i)}$ (a total of 6 features, one for each HMM)
Model	Sum prob.	$\sum_{i \in S} p_i)$-a total of 6 features, one for each HMM
Model	Universal	The probability of the *Universal state*
Data	Acceleration	Instant magnitude of acceleration: $\sqrt{a_x^2 + a_y^2 + a_z^2}$
Data	Gravity	The value of X component of gravity
Time	Elapse time	For each HMM, the time elapsed since state 0 had the highest probability –a total of 6 features

Table 4.2: Features used for early gesture classification (for the 6 VTT gestures)

observation sample $O(t)$, we compute the following features, : (i) sum of the probabilities of each of its hidden states–i.e., $\sum_{i \in S} p_i)$; (ii) normalized expected value of the hidden state–i.e., $\frac{\sum_{i \in S} p_i * s_i}{\sum_{i \in S} p_i}$. Given N gestures, this leads to a $2N$-element vector of state-related features. In addition, we also consider the following additional features: (iii) the probability of the *Universal* state; (iv) the magnitude of the acceleration data (i.e., $\sqrt{a_x^2 + a_y^2 + a_z^2}$, where a_i represents the acceleration value along the i^{th} axis); (v) the X-component of the gravity vector (as this is an indirect indicator of the hand's orientation); and (vi) a vector $TS = [ts_1, \ldots, ts_N]$, where the i^{th} element expresses the time elapsed since the initial state (s_0) in the i^{th} HMM had the highest probability among all its hidden states. The acceleration feature helps distinguish deliberate strokes (which typically have higher acceleration) from other 'random' gestures, while the vector TS effectively guards against strokes being recognized too early (especially for the 'push' and 'drive' gestures).

Table 4.2 summarizes these features. This set of features is then used by a Random Forest classifier (trained in a supervised fashion) to determine the most likely gesture, for each incoming observation sample $O(t)$. Initially, the probability of the *Universal State* is the highest, and the classifier indicates that the inertial data (observed till this point) is most likely to be a 'non-gesture'. The classifier automatically outputs a gesture label, once sufficient data has arrived.

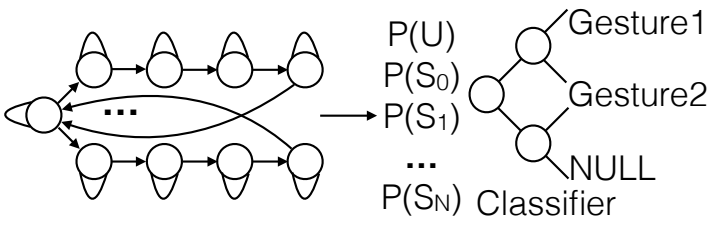


Figure 4.10: A classifier is used to classify gestures, using probabilities of states as features

4.5 Experimental Results on Early Gesture Detection

We first evaluate the efficacy of our proposed stream-compatible, early gesture detection technique using a *personalized* model–i.e., where a separate HMM (per gesture) is trained for each individual. This is similar to many prior studies on gesture recognition [109, 96, 80] that have also utilized personalized HMM models. The evaluation is done using the 'leave one gesture out' (LOGO) method. Because our system includes 2 stages: the HMM and the classification, we apply 2 training passes. At the first pass, we train the HMM model on the $N-1$ instances of each gesture (N is the total number of each gesture). We then feed the stream of the entire dataset into the HMM model. The features extracted from the training part (containing $N-1$ instances of gesture and non-gesture) are stored and used as training data for the downstream classifier (Random Forest). The features extracted from the testing part (containing the test gesture and non-gestures) are used to test the classifier. Of course, our classifier produces a label for every sensor sample. To make the classification more efficient, the *Universal State* is used to pre-filter the result. If the *Universal State* has the highest probability or the most probable gesture has not reached a state threshold (empirically chosen as 3), non-gesture is assigned; otherwise, the downstream classifier classifies the candidate gesture. Whenever the classifier declares a gesture *A*, our system considers the current and subsequent sensor sample as gesture *A* until the probability of the corresponding HMM state se-

quence drops below that of the *Universal State*.

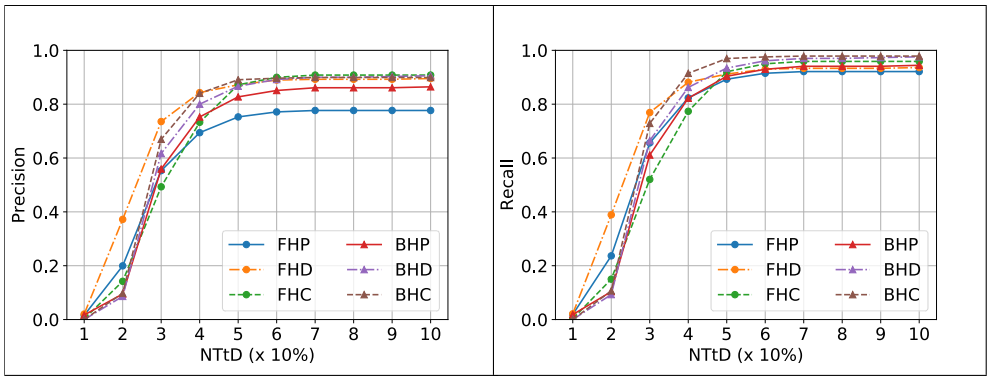
We sum up the total true positives, false positives, false negatives in each class across all participants; subsequently, we compute the average precision and recall values based on these numbers.

4.5.1 Accuracy vs. Fraction of Gesture Completed

Because our principal goal is to understand the early gesture detection capability, we compute the precision and recall of our model as a function of different *fraction of the total gesture duration*. More precisely, we compute the gesture detection accuracy for various values of *Normalized Time To Detect (NTtD)*[58] (in increments of 10%), which is the portion of gesture that has been completed. Figures 4.11 illustrate the overall precision and recall for each of the 6 strokes (gestures) as a function of the NTtD value, with the first 2 figures corresponding to study 1 (novice/inexperienced users) and the latter 2 figures corresponding to study 2 (experienced users). We see that:

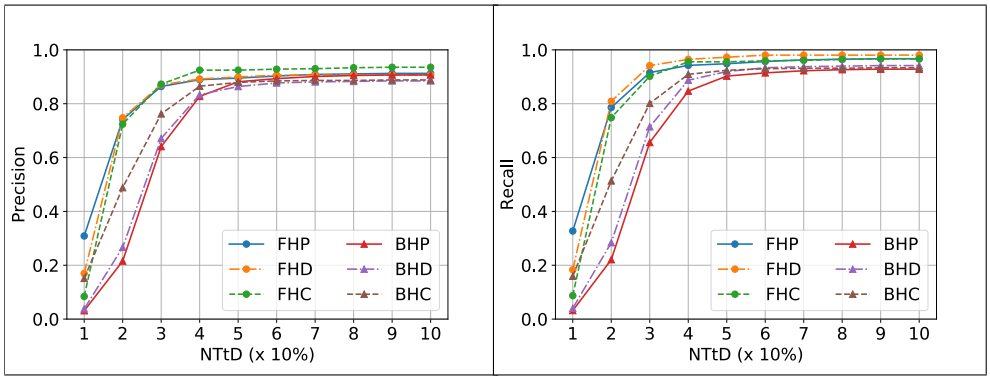
- *Early Detection:* At NTtD values of 0.5 (i.e., at about 50% of the gesture), both precision and recall are higher than $\sim 84\%$ for all the strokes. Moreover, after the NTtD value of 0.5, all the precision and recall values do not change considerably with an average precision of 87.4% and average recall of 95.2% (for inexperienced users).

- *Better Performance under Stable Gestures:* We see that, at identical NTtD values, the gesture recognition performance is superior for experienced users, compared to inexperienced users. In particular, at NTtD=30%, the precision & recall for users in study 2 is 64-87% and 65-94% respectively, while the comparative precision & recall is between 50-73% and 52-76%, respectively, for study 1. Clearly, experienced users exhibit more consistent stroke-making, implying that even a modest 30% samples of testing gesture are enough, whereas novice users tend to exhibit higher variability



(a) Dataset1 Precision (Inexperienced)

(b) Dataset1 Recall (Inexperienced)



(c) Dataset2 Precision (Experienced)

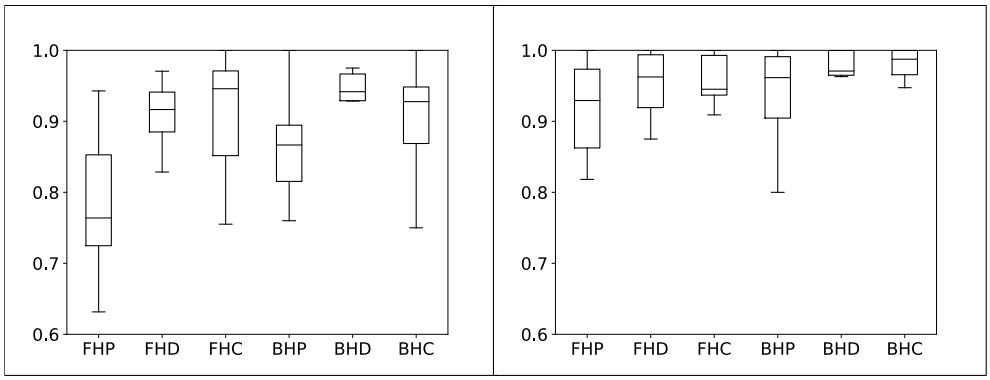
(d) Dataset2 Recall (Experienced)

Figure 4.11: *Left*: Precision and *Right*: Recall w.r.t. Normalized Time To Detect (Across Both Studies)

in performing the same stroke multiple times.

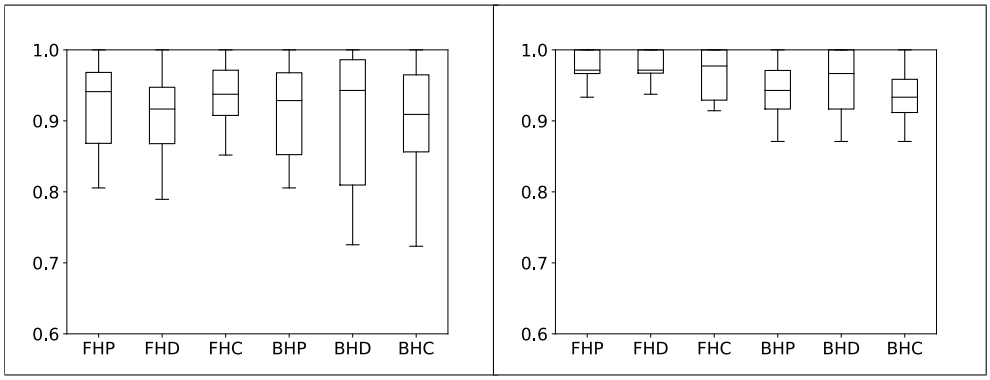
Variation Across Users:

We observed that even person-specific models show significant differences in the cross-user variation in gesture recognition accuracy, for different gestures. Figures 4.12 plots the variation in the precision and recall, respectively, of gesture detection for each of the 6 VTT gestures, for both Study 1 and Study 2. (This is in contrast to Figures 4.11, which plot the *average* values, across all individuals.) We see that the median precision values (almost always above 87% for Study 1, and above 90% for study 2) and recall (above approx. 95% for study 1 and 95% for study 2) values are fairly high. However, there was one individual user (in study 1) who exhibited significantly lower accuracy, due to the specific individual's inability to play the 6 strokes distinctly. In general, the



(a) Dataset1 Precision (Inexperienced)

(b) Dataset1 Recall (Inexperienced)



(c) Dataset2 Precision (Experienced)

(d) Dataset2 Recall (Experienced)

Figure 4.12: Box plot of *Left*: precision and *Right*: recall across users. X-axis corresponds to the 6 distinct gestures.

variation across experienced users (Dataset 2) is significantly smaller than for inexperienced users (Dataset 1).

4.5.2 The Utility of the Classifier

For early gesture detection, our approach combines the streaming-oriented HMM with an HMM-state driven classifier. To understand the importance of the classifier, we studied the accuracy of gesture recognition both with and *without* the final HMM-based classifier. In general, HMM-based models can use a threshold (on the confidence value) to classify gestures [96, 109]. Similarly, we can classify gestures directly using our modified HMM model, where the *Universal State* serves as a dynamic threshold. Moreover, our model produces confidence value for every sample, as well as the most probable current state. Intuitively, we would choose the gesture that evolves to the higher state (a per-

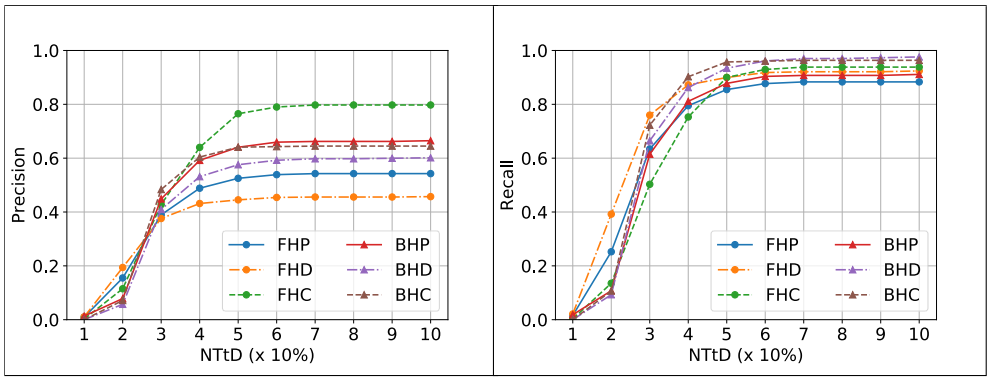
fect gesture would, in fact, evolve to the final state). However, waiting for the evolution to the final state incurs latency, and we cannot perform early detection of a gesture. On the other hand, if the system declares a gesture too early, it is more likely to produce a false positive.

A little reflection will show that determining the threshold is the main problem with the threshold method. Our addition of a classifier is intended precisely to tackle this conundrum and to reinforce the decision making of the system. In particular, the classifier observes the states, probabilities and gesture labels of every sample of training data, and can thus avoid recognizing a gesture under isolated samples of spurious observations. For example, in certain situations, a non-gesture may manifest in a high (close to final) HMM state and probability of *Forehand push* gesture; however, at that point, the hand may be pointing upward (x-axis value of gravity is high), making a *Forehand push* very unlikely. By using features corresponding to different evolutionary points of the gesture, our gesture recognizer becomes more *robust*.

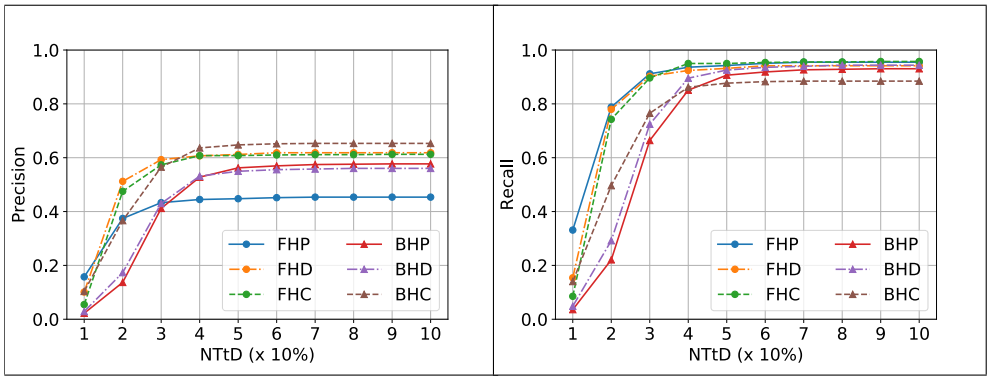
Figure 4.13 demonstrates that the subsequent Random Forest classifier significantly improves the recognition accuracy. Without the classifier, the accuracy is low because many non-gestures can evolve to the equally high state and probability. The classifier eliminate those false positives by looking at the orientation of the wrist and the strength of the gesture. We see that our modified HMM continues to be effective in detecting gestures correctly and early.

4.5.3 Performance of Person-Independent Models

To additionally understand the need for such personalized models, we also study the overall gesture recognition perform for a *person-independent* model. Note that, in general, the accuracy of person-independent gesture recognition algorithms is usually quite low–e.g., Thomaz et. al. [121] reported low values for precision and recall when detecting 'eating' gestures using a person-independent model.

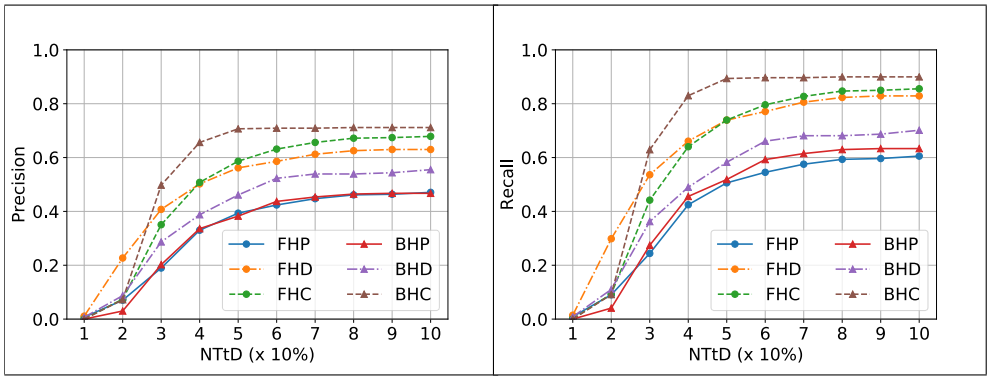


(a) Dataset1 Precision (Inexperienced) (b) Dataset1 Recall (Inexperienced)



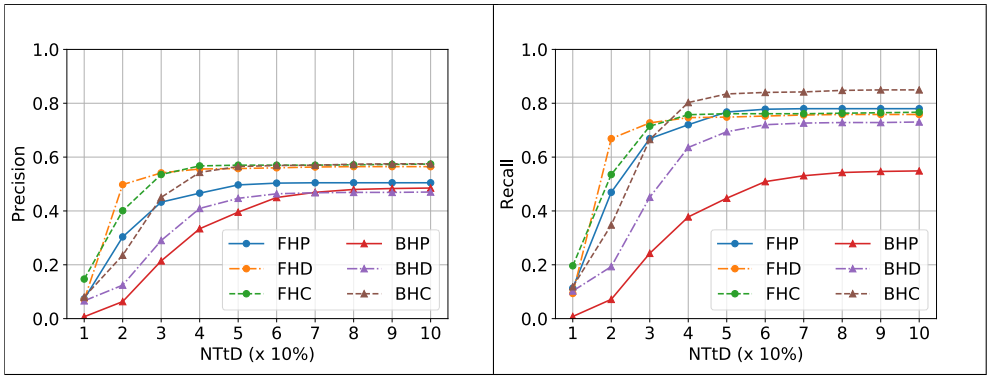
(c) Dataset2 Precision (Experienced) (d) Dataset2 Recall (Experienced)

Figure 4.13: Early detection ability of our stream-HMM model. Left: without the subsequent classifier. Right:with the subsequent classifier. probabilities.



(a) Dataset1 Precision (Inexperienced)

(b) Dataset1 Recall (Inexperienced)



(c) Dataset2 Precision (Experienced)

(d) Dataset2 Recall (Experienced)

Figure 4.14: Precision w.r.t. Normalized Time To Detect (Person-Independent Model).

We use the Leave One Person Out (LOPO) validation technique, whereby we train the model using $K - 1$ participants and test the system with the other participant, repeating this process for every distinct participant. As before, we measure the precision (See Figure 4.14a) and recall (See Figure 4.14b) as a function of NTtD, with a step of 10%. As expected, the person-independent model gives rise to significantly lower accuracy. However, even then, we see that the early detection capability of the gestural model is relatively unaffected– e.g., for an NTtD value of 0.6 (i.e., at the time instant when a gesture is 60% complete), the recall of the person-independent model is higher than 50% for all gestures.

This LOPO validation shows that the development of a generalized model may not be impossible. All gestures show quite high recall and the precision is around 50%. The two gestures *Forehand Push* and *Backhand Push* have the

lowest precision and recall in the study 1 (inexperienced players). As we mentioned before, the participants in this study are not professional players, so they usually play these two gestures very similarly to either *Drive* or *Chop* gestures. Even in study 2 (experienced users), the *Backhand Push* has significantly lower recall compared to other gestures. These results suggest that more sophisticated classification models (e.g., the Deep Learning-based LSTM models) may indeed offer more robust, person-independent classification. However, we do not pursue this investigation further in this thesis, as the development of "improved classifiers" is not the core focus.

4.5.4 Comparison with E-Gesture Baseline

To evaluate how well our method can recognize gestures early, we compare the precision and recall of our proposed approach with a competitive baseline. As described in Section 4.4.1, the baseline is based on a modified version of the segmentation-cum-HMM model of [96], which we showed to provide better results. This baseline classifier operates on the data of each segment and outputs the gesture with the highest confidence value at *any* intermediate point of the gesture.

Figure 4.15 plots this comparative performance–with the horizontal line showing the precision and recall values for the baseline (which operates on the entire segment and thus does not vary with NTtD values). The figure shows that our proposed approach matches the precision and recall values of the baseline at an NTtD value of 0.4 (the precision is 87.0% compared to 88.0% of the baseline, and the recall is 92.0% compared to 88.0% of the baseline), implying *that we are able to detect gestures as accurately as the baseline (which needs to wait for the entire gesture segment) much earlier, using only the first 40% of a gesture*.

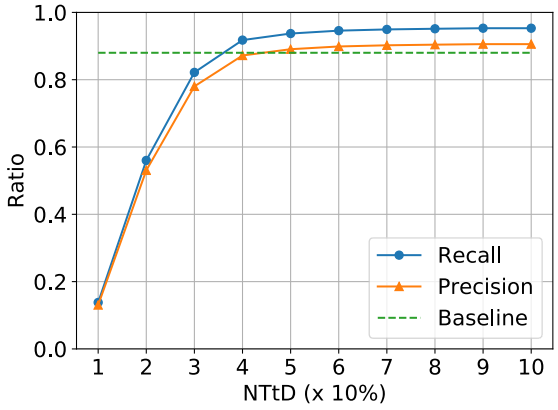


Figure 4.15: Early detection capability of proposed method vs. enhanced E-Gesture baseline.

4.6 Gesture-State-Enabled Trajectory Tracking

While many studies rely on hand tracking to recognize gestures, hand tracking using sensors in a wearable device is not trivial because of the noise in sensory data. In this work, we explore the reverse problem: *Use knowledge of the gesture being performed to improve the accuracy of hand tracking*. This is based on the observation that during gestures of a specific type, a user's hand is likely to follow a more-limited "trajectory cone" in the 3-D space. In particular, if we can detect gestures accurately and early (which we've demonstrated in Section 4.4), we are likely to be able to estimate the position of user's hand based on the instant orientation of the hand more accurately.

4.6.1 Existing Approaches of Hand Tracking

A widely-used prior approach is to use dead-reckoning over the inertial sensing data. As we can extract the angle of the hand and its acceleration, we can intuitively apply the integral to calculate the 3D positions of the hand, assuming that we can obtain the initial reference position of the hand. However, this method often suffers from accumulated errors caused by noises in acceleration and orientation sensor readings. In addition, it is not easy to reset the reference point to offset the integral calculation once the table tennis play is started; note the

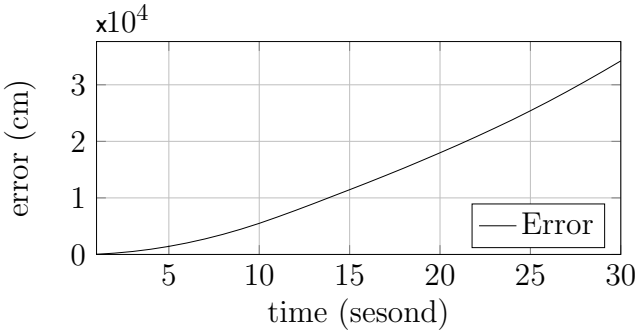


Figure 4.16: Tracking error of the dead-reckoning approach.

periodic reset of the reference point is essential for reckoning-based approaches to reset the accumulating error. We implemented and evaluated this method over our dataset described in Section 4.3. Figure 4.16 shows the trajectory tracking errors as the time progresses. The tracking error increased to 300 meters after just 30 seconds, making the approach infeasible for our scenario.

ArmTrack [113] adopted a different approach to address this problem of error accumulation. Its key idea is to use a pre-built look-up table (LUT) that maps an orientation of the hand in a particular position to a specific 3-D location; in particular, they consider the possible angles of the shoulder and elbow joints to decide the mapping. This approach works well under the assumption that the user's shoulder is fixed. However, this method does not correspond well with our scenario, as table tennis interactions have complex kinematics and usually involve significant body movement during a game session. For the same stroke, the spine orientation, as well as the angles of the shoulder and elbow, could be highly variable, making it difficult to build an accurate look-up table. Moreover, for a VR-based game such as *VTT*, the wrist position should be computed, not relative to the body, but relative to the "virtual world" coordinates. For example, a user starting *VTT* on her VR display will see the board straight ahead, but as she moves her body, the location of her hand should shift, relative to the table (and thus the trajectory of an incoming ball), even if the hand doesn't move relative to the body. Accordingly, in our approach, (1) we assume the 'virtual world' coordinates to be defined by the user's orientation at the start of the

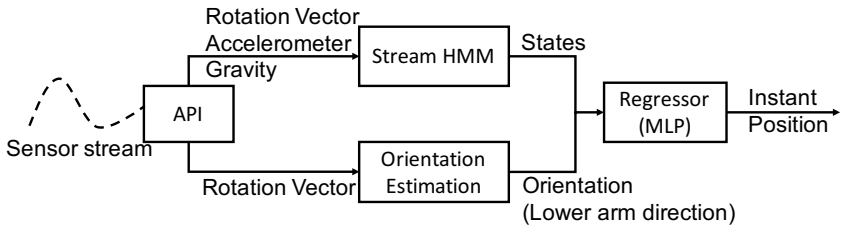


Figure 4.17: The overall logic of trajectory tracking

game; (2) during subsequent game-play, we first compute the hand trajectory relative to the body (the user's forehead); (3) we separately track the forehead's motion, relative to the virtual world coordinates, using a separate head-mounted device (e.g., a smartphone mounted on a VR device such as Samsung Gear VR); and (4) finally, utilize the forehead motion vector to translate the hand's body-coordinates to the virtual world coordinates.

4.6.2 Gesture-State-Enabled Trajectory Tracking

To enable accurate, real-time hand tracking, we proposed a new method called *Gesture-state-enabled Trajectory Tracking*. The core idea behind this method is to utilize the intermediate states of the gesture's progress, along with the orientation of the hand. As described in Section 4.4, our gesture recognizer continuously outputs which gesture the player is likely performing as the gesture progresses. When the information of the gesture progress is combined with the hand orientation, it is possible to quickly narrow down the possible positions of the hand to a smaller, *plausible* area.

Figure 4.17 illustrates the trajectory tracking logic. As the first step, it computes two different features upon a sensor reading:

- *Orientation of the hand:* the trajectory tracker first computes the orientation of the hand, more specifically the orientation of the wrist-elbow limb. This can be calculated by computing the X-axis unit vector (1.0, 0.0, 0.0) of the smartwatch in the reference coordinate system (i.e., the coordinate system at the start of the game, when the table tennis board is directly in

front of the user). The use of the x-axis component is important as it is least affected by the wrist rotation; note that table tennis gestures involve wide rotations of the wrist, and thus, y-axis and z-axis components can vary considerably even at the same position of the hand.

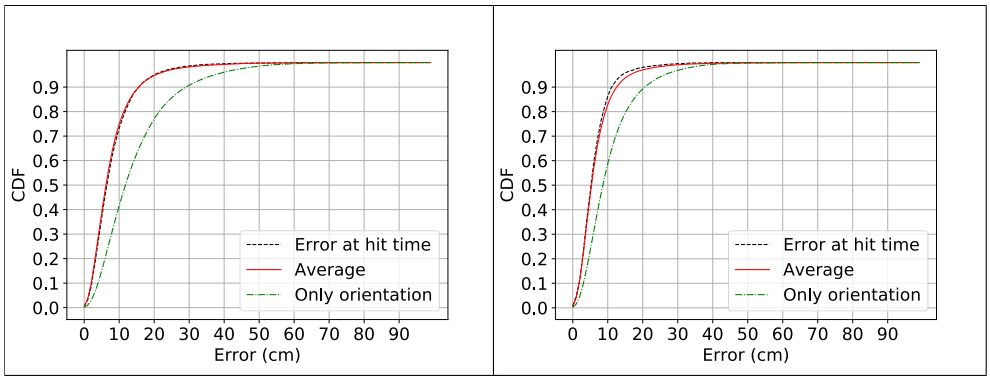
- *Current state of the gesture:* it retrieves the current HMM states of the gesture, which indicates how much progress has been made for the gesture being currently performed. We still utilize the average states and probabilities which are described in section 4.4 as features to estimate the wrist position.

These two input values are streamed into a regression model, named *Multilayer Perceptron (MLP)*, which computes the possible location of the hand. We use a fully connected MLP (in Weka machine learning library [129]) with 3 layers of hidden units. Number of nodes in the 3 hidden layers are empirically set to 9, 18 and 9 correspondingly.The model is pre-trained with the hand trajectory data (computed using the "ground truth" video data of a player captured by the ZED camera). As mentioned before, the hand trajectory is defined relative to the 'virtual game' coordinates– this positional ground-truth is measured (and used in training the MLP) by tracking the user's head location, and adding the head-to-hand vector to this location.

4.6.3 Hand Tracking Performance

We evaluate the system using 10-fold cross validation (each gesture instance is a primitive unit). The data in this experiment is generated from the gesture recognition experiment. The states and wrist orientation of each test gesture are combined with the corresponding position ground truth to create a new hand tracking dataset.

Figure 4.18 shows the CDF of the average hand tracking error (the average computed over all points of a gesture instance), based on a personalized model.

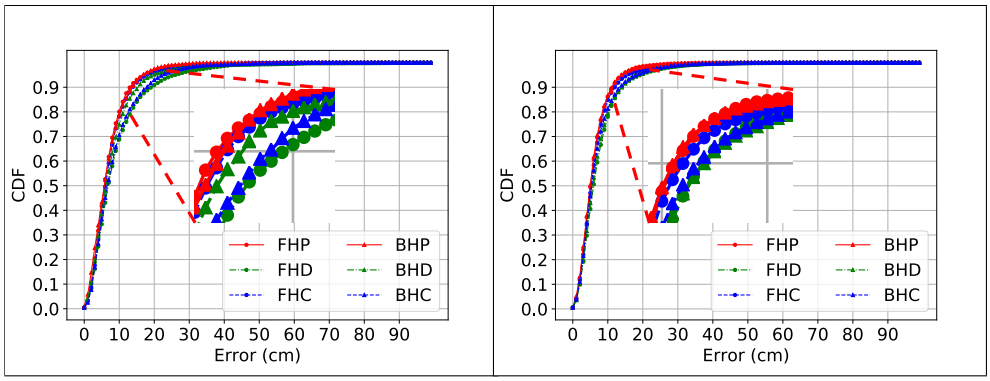


(a) Dataset1 (Inexperienced) (b) Dataset2 (Experienced)

Figure 4.18: CDF of Average (and Hit Time) Trajectory Tracking Error.

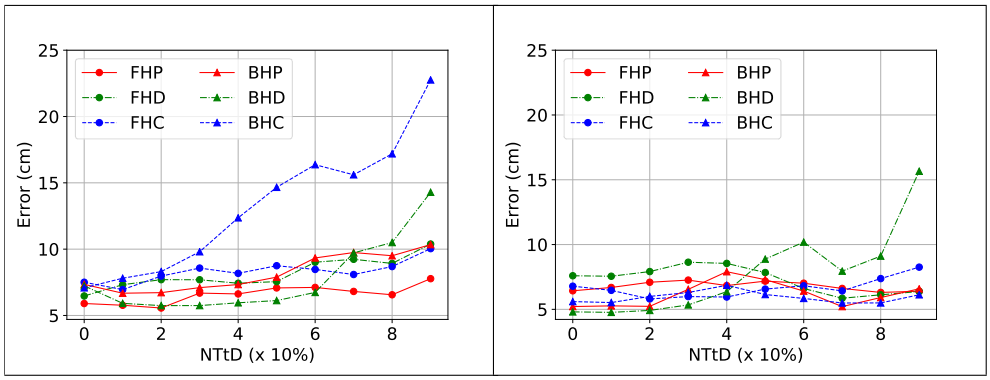
We compare the tracking error of our approach with a baseline (named "only-orientation") that performs regression, but without using the intermediate state of HMM. In addition, we also plot the CDF of the errors, computed only at the 'hit time' (when the bat hits the ball). While the baseline model achieves a median error of 9.5cm, our approach improves the tracking error significantly with a median error of 6.3 cm (for experienced users). The error at hit time (6.2cm) is slightly lower (but not by a significant amount) compared to the average error. As anticipated, the experiment with inexperienced users shows a slightly higher median error of 7.2cm. We can see a significant improvement (around 15-30cm) in tracking accuracy over the baseline at the 90^{th} percentile. The results show that using intermediate HMM-state information helps the regression model to estimate wrist position more accurately.

We further break down the tracking error per gesture type. Figure 4.19 shows the CDF of the average tracking error, for each of the 6 strokes. In this experiment (with experienced players), we observe a lower difference in the tracking error, across different gestures, for experienced users. However, at 90th percentile, we still see higher error values for the *Forehand Drive*, *Backhand Drive* and *Backhand Chop* gestures. The two *Drive* gestures are sometimes occluded in our training data because the player moves the racket towards the camera during the gesture. In these cases, the interpolated ground-truth is used. The *Backhand Chop* also has the higher error, but less than the two *Drive* gestures.



(a) Dataset1 (Inexperienced) (b) Dataset2 (Experienced)

Figure 4.19: Stroke-specific tracking error distribution using (a): inexperienced user dataset, and (b): experienced user dataset.



(a) Dataset1 (Inexperienced) (b) Dataset2 (Experienced)

Figure 4.20: Tracking error distribution vs. gesture progress, using (a): inexperienced user dataset, and (b): experienced user dataset.

Both *Drive* gestures and *Chop* gestures usually involved fast movements of the arm, as a result of which the orientation and ground truth estimators are less accurate, compared to the slower hand movements in the other strokes.

We further investigate how the tracking error change as a gesture progresses. Figure 4.20 shows the tracking error as a function of the percentage progress of the gesture. Contrary to our expectation, there appears to be no discernible trend relating the tracking error to the progress of the gesture. There is an apparent significant increase in the error of *Backhand Drive* or *Backhand Chop* towards the end–however, note that these two gestures are often occluded, as a result of which the ground-truth samples of the hand's location are sometimes missing or not as accurate as the other strokes.

4.7 User Perceptual Experience

The previous sections have demonstrated that our approach can recognize gestures early and accurately (with over 90% precision by the mid-point of a *VTT* gesture) and that it can then track the hand's trajectory, in real time, with an error of no more than 6-8cm. Clearly, such errors in tracking will manifest themselves in errors in estimating the *hit time*–i.e., when the racket makes contact with the ball. For a system such as *VTT* to be used, users must have the *perception* that the virtual experience faithfully recreates the "real world".

To study this perceptual effect, we recruited 5 players who are students in our university. We suspended a table tennis ball in the experiment room; each participant adjusted their body position, as well as the ball's height, such that they are comfortable in striking the suspended ball using the Forehand Push or Backhand Push strokes. Each participant wore a smartwatch (on their stroke-playing hand), which was instrumented with our gesture recognition software. The software was provided with the ball's *ground truth* location (computed using the Zed camera). The participant was then required to try to hit the ball with 25 Forehand Push shots and 25 Backhand Push shots. Whenever the smartwatch detected a gesture, it estimated the position of the wrist. If the estimated wrist position is within 8cm (which is about the diameter of the bat) of the suspended ball's position, then the system will emit a "beep" sound. We then asked the participant if they perceived the "beep" sound to be before, later or at the true *hit time* (the time when the user saw his hand actually hit the suspended ball). To avoid the need for building a personalized gesture model (which would have required additional training data from each user), we instead used a previously existing gesture recognition model derived from a participant of the prior studies. As expected, as this is a model trained on a different user, the gesture recognition accuracy is a bit lower than the results reported earlier (for personalized models).

As mentioned before, the user study measures users' perception of system

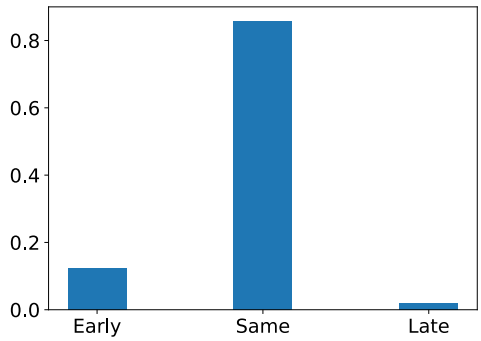


Figure 4.21: User's perception of our system response time.

response time. Figure 4.21 summarizes the users' response to our query: we see that for $\approx 85.7\%$ of the gesture instances, users felt the "beep" sound happened concurrently with the true *hit time*. To further understand the likely usability of a system built using our technology, each user was asked (once each after the forehand and backhand sessions) about the "usability" of a future *VTT* training system built using this gesture recognition technology. The user rated the usability of the technique on a scale from varying from 1(unusable)-5 (absolutely usable). The average score across 5 users (based on sessions where the gesture recognizer was most accurate) is 4.2 out of 5. This result gives us confidence that our gesture recognition and hand tracking system can indeed be embedded in useful interactive applications (such as *VTT*).

4.8 Discussion

Our experimental results show that our system outperforms competitive baselines by a significant margin–both in terms of the gesture detection latency and the 3-D hand tracking accuracy. However, there are several additional questions and issues that we must consider.

Diversity of dataset: Currently I have studied early gesture recognition and hand tracking with 6 table tennis gestures. It is still unclear how well the same technique performs with other types of gestures. In addition, the dataset in-

cludes only right-handed adults. A more diverse dataset (e.g, including children, elderly) may reveal other person-specific factors which affect the performance of the proposed technique.

Limitation of early gesture recognition technique: Our current approach for early detection is based on the features derived from the HMM state probabilities. However, the speed of the gestures among players may vary considerably across expertise level (even within the same level). We may need to consider features that can represent the gesture speed and/or clustering the data before recognizing gestures. Moreover, the early detection approach implicitly assumes that the gestures are prefix-free–i.e., one gesture isn't a sub-trajectory of another gesture. If the gestures are not prefix-free, then the observation interval will need to be larger than the duration of the smallest prefix, to help disambiguate between multiple gestures. Finally, though our proposed technique have high accuracy with personalized model, the accuracy with person-independent model is much lower. Perhaps the HMM model is not able to model all the essential person-independent relations. Newer machine learning techniques for sequential data such as Long-Short-Term-Memory might be able to address the problem by automatically extracting features from the data, but the processing time and power consumption of the system might increase respectively.

Limitation of hand tracking technique: The current experimental study focused primarily on determining the trajectory of the hand, under the assumption that the body position and orientation was relatively unchanged. In applications where a user needs to change his/her body orientation continuously, such as dancing, our technique may fail to estimate wrist position. For this type of applications, the system will need to fuse the body orientation together with the smartwatch orientation to get improved 3-D tracking of hand movement. In particular, the gesture detection logic is currently based on features that are computed in the earth's reference frame, whereas the actual gesture trajectory may vary depending on the body orientation. Accordingly, accurate gesture

recognition itself may require fusing of the hand orientation with the body orientation/posture data (e.g., obtained from the smartglasses)–this may be difficult to execute at the desired low latency.

Need for sensors with high sampling rate: our current work and results utilized multiple sensors (accelerometer, gyroscope and magnetometer) for hand tracking and also implicitly used high sampling rates (100Hz for accelerometer, gyroscope and magnetometer). While our results show the efficacy in improving early gesture recognition and hand tracking, the resulting energy profile may be too expensive for execution on highly battery-constrained platforms (e.g., the battery-less *WiWear* platform presented in the previous chapter). Accordingly, we will need to additionally evaluate the accuracy vs. sensing energy tradeoffs (e.g., by using the sensor sampling frequency as a 'control knob') to understand the regime of applicability for our technique to such battery-less devices. I shall discuss this issue further, in Chapter 5.

4.9 Reflections and Lessons Learned

During the development and the experiments for this Chapter, I have learned several lessons which might benefits other students who plan to work in this area of research.

Having step-by-step printed script for an experiment: Usually an experiment is an expensive process in both time and finance. Redoing an experiment session with one participant is really difficult, and sometime, invalid. Every mistake during an experiment is costly, and usually results in losing the data of that participant. An experiment usually consists of many ordered steps. Every change in these steps may result in a failure. So it is important to follow these steps exactly for every participant. If I do not have a step-by-step printed script, it is very easy to forget one step or take one step in wrong order even if I remember the experiment procedure by heart. The reason could be another participant

is waiting for his session while the first session is over due.

Managing participant's data is not simple: It seems very simple to store experiment data in a computer. However, when there are many participants for an experiment, it completely matters how the data is stored. Especially, if the data is stored under participants' encoded name (e.g, p1, p2 etc.), and there are more than one type of data to be stored, it is extremely confusing later when there are so many files and folders with the same name. Another problem is that, smartphones and smartwatches are commonly used to collect data. In many cases, the data must be buffered locally before it is stored in a secure place. A common problem is that some unexpected events happen during a session and the session must be restarted. Later we might loose track of which is the correct data to be used. So it is important to store the data of each participant in an identifiable way immediately after the experiment session before starting any other experiment session.

Chapter 5

Feasibility Analysis: Early Gesture Recognition and Tracking for Battery-less Devices

In Chapter 3, I presented our proposed *WiWear* framework to enable motion sensing on a battery-less wearable which harvests energy from beam-formed WiFi transmissions. In Chapter 4, I detailed our method of early gesture recognition and tracking which enable low-latency (and also low-complexity) recognition of gesture, and accurate hand trajectories tracking. However, it is still unclear if it is feasible to deploy such a low-complexity motion sensing model on a *WiWear* device to support the motivating scenarios mentioned in Chapter 1. Of course, one can expect better accuracy with higher fidelity of motion sensor data. However, higher fidelity also results in higher power consumption. To answer the research question: Is it feasible to achieve low-latency motion sensing using battery-less wearables?, I shall analyse to what extent of fidelity a *WiWear* wearable supports low-latency motion sensing.

In this chapter, I shall investigate whether the early gesture recognition and techniques, described in the previous chapter, are amenable to execution in tandem with the *WiWear* battery-less wearable devices previously described. The

key consideration is, of course, the energy overhead–i.e., to determine if the energy consumption of such algorithms (including the sensor sampling, computation and communication overheads) fits within the energy-harvesting envelope of the *WiWear* platform. More specifically, as the energy overhead is known to be affected by the sampling frequency, I shall study how the accuracy of the gesture recognition technique varies with such sampling frequencies, and thereby establish the salient energy-vs-accuracy tradeoffs. Another factor, which affects the energy consumption of the device if the device performs the processing task, is the complexity of the processing pipeline. Based on such analyses, I shall identify the operating regimes (and thereby the gesture-based use cases) that a *WiWear*-like device can support.

Table 5.1: Analysis methods of different parts in the feasibility study.

Part	**Analysis Method**	**Data**	**Sub-section**
Accuracy vs. sampling rate	re-training models	sub-sampled real-world data	5.1
Accuracy vs. complexity	re-training models	original real-world data	5.1
Power consumption vs. sampling rate	simulation	data-sheets, model complexity	5.2.1, 5.2.2
Power consumption vs. complexity	simulation	data-sheets, model complexity	5.2.1
Operational lifetime	simulation	data-sheets, model complexity, simulated harvested energy	5.2.3

Due to the lack of support for machine learning libraries on low-power micro-controller (STM32L053) used in the *WiWear* prototype, the actual implementation of our early gesture recognition and hand tracking on a *WiWear* wearable is not possible. Therefore, the power consumption of components are analyzed using synthetic data; whereas the accuracy of algorithms are evaluated using sub-sampling of real-world data. To provide the reader an easy summary of the evaluation strategies, Table 5.1 summarizes relevant aspects of the analy-

ses reported in different sections of this chapter.

5.1 Effects of Varying Sampling Rates and Model Complexity on Sensing Accuracy

Though higher sampling rate usually results in better sensing fidelity, it also drains more energy. However, the impacts of the of varying the sensor sampling rate differ depending on the underlying application. This is a trade-off that can be exploited to achieve an optimal operating condition in which the sensing fidelity is sufficiently high and the energy consumption (by the sensors) is low enough to be compatible with the limited amount of energy harvested by *WiWear* wearables. To explore different sampling rates on the dataset I have recorded from Table Tennis players with a sampling rate of 100Hz, the dataset is first sub-sampled at different rates (50Hz, 25Hz, 12.5Hz, 6.25Hz). Then, it is interpolated again to 100Hz so that the same model can be applied to different sampling rates. Depending on the frequency spectrum of the original data, the sub-sampled data may lose a certain essential frequency components. Figure 5.1 shows the frequency spectrum of linear acceleration at different sampling rates. Figure 5.1.a shows two main frequency components of 1.6Hz (very close to the DC component) and 4.6Hz (the second peak). Figure 5.1.e shows that the sub-sampled data at 6.25Hz loses an essential component of 4.6Hz (the second peak has disappeared). The sub-sampled data at 25Hz and 12.5Hz show some frequency losses, but can preserve the 2 strongest components. Figure 5.2 shows the reconstruction loss of linear acceleration in root mean square error (RMSE). As expected, the reconstruction loss of re-sampled data increases dramatically at 6.25Hz to more than $1.25 m/s^2$ (the mean value of linear acceleration during gestures is $8.2 m/s^2$).

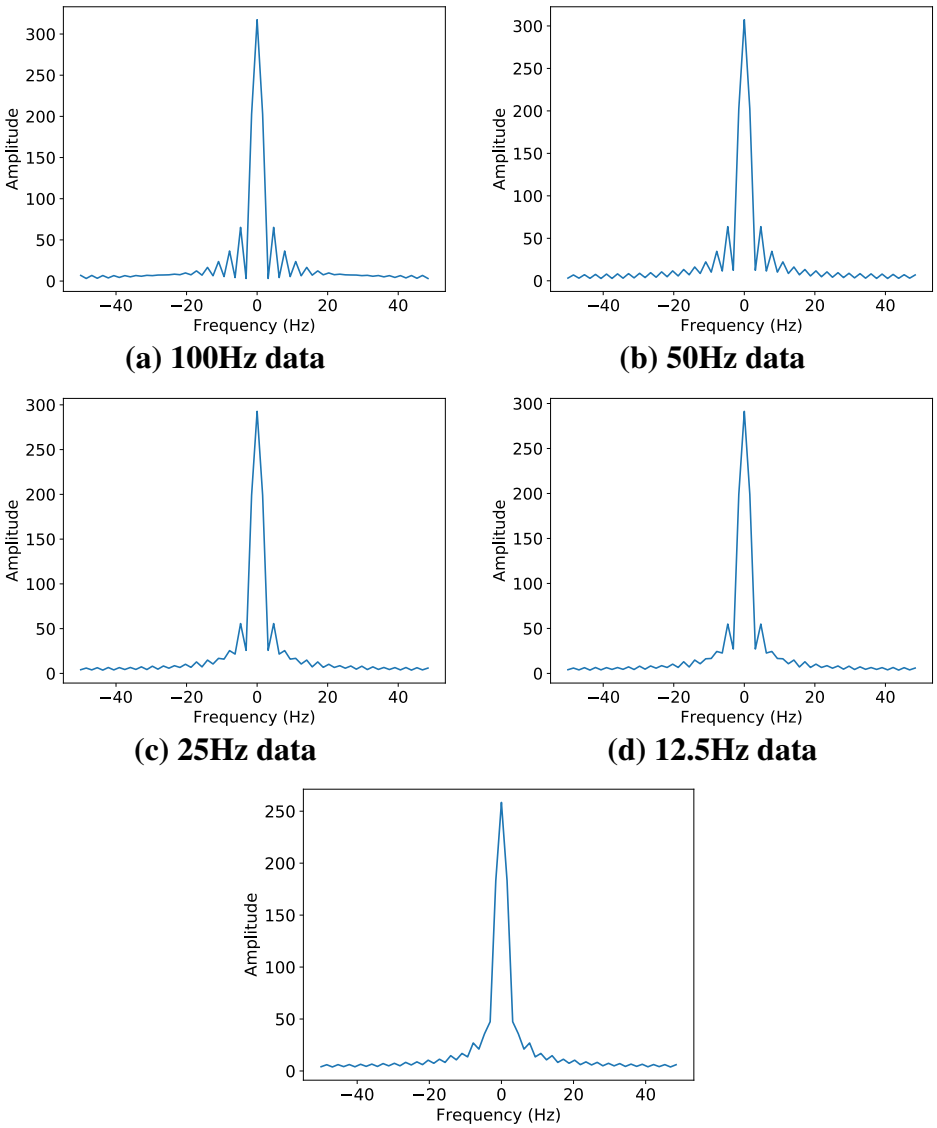


Figure 5.1: Frequency spectrum of linear acceleration during a forehand push gesture at different sampling rates.

Specifically for the early gesture recognition and tracking, which I have described in Chapter 4, the Figure 5.3 shows that the system can achieve an F1 score of 90% with a sampling rate of 25Hz compared to 93% with a sampling rate of 100Hz. Even at 12.5Hz, the system can still achieve a fairly high F1 score of 86%, but then the achieved accuracy drops significantly down to only 55% when the sampling rate drops down to 6.25Hz. The fact that the same model was kept across different sampling rates suggests that our early gesture recognition is robust to sampling rate change (e.g, the system can dynamically

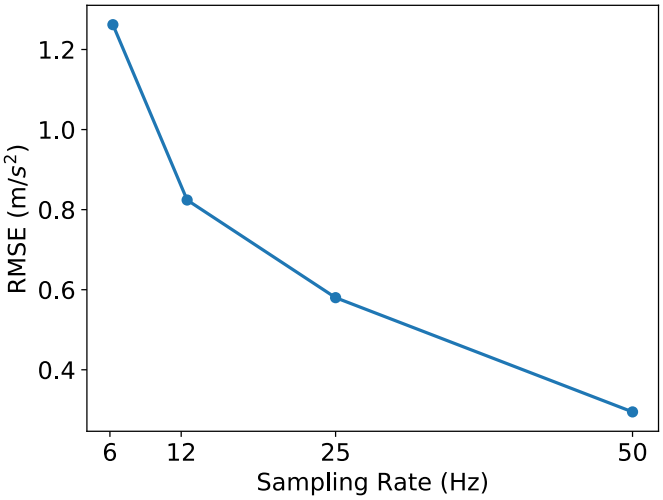


Figure 5.2: Reconstruction loss of re-sampled data at different sampling rates (Hz)

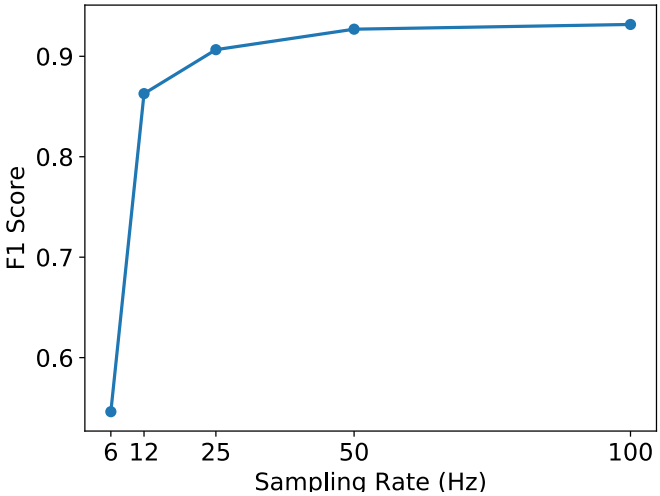


Figure 5.3: F1 score of our early gesture recognition at different sampling rates.

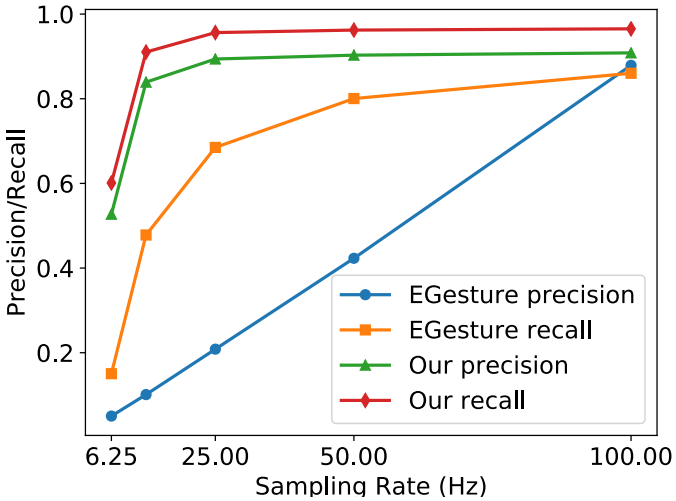


Figure 5.4: Precision and recall of our method compared with a baseline (E-Gesture).

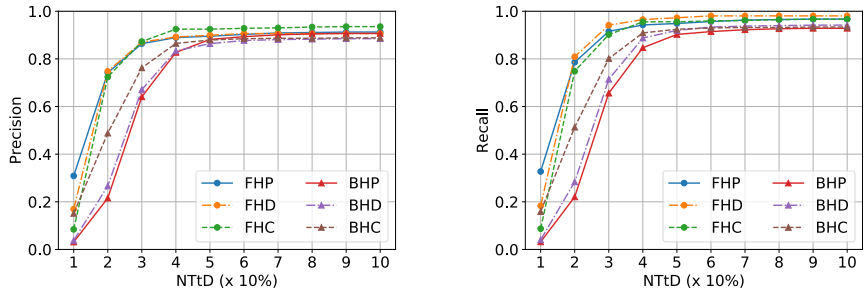
change its sensor sampling rate to adapt to dynamic changes in the available energy budget).

Equally importantly, our gesture recognition and tracking technique is more robust, than other baseline alternatives, to varying sampling rates. As shown in Figure 5.4, which compares our method with a baseline based on an implementation of EGesture [96]. EGesture precision drops almost linearly when sampling rate drops from 100Hz down to 6.25Hz. The precision drops much faster than recall. It means that, at lower sampling rates, EGesture segmentation proposes too many spurious segments, and the subsequent threshold-based HMM technique is not robust enough to filter out such spurious data. On the contrary, our method still maintains fairly high precision and accuracy even when the sampling rate drops down to 25Hz.

5.1.1 Early gesture recognition

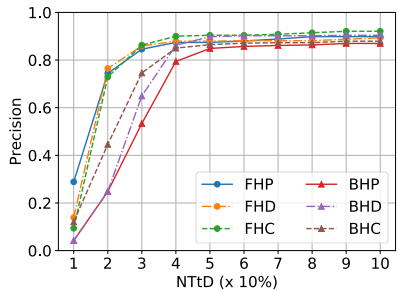
Early gesture recognition under different sampling rates

Figure 5.5 plots the precision of gesture recognition for different strokes, as a function of the fraction of completed gesture (NTtD), for different sampling rates. We see that the ability to perform *early* gesture recognition is not signif-

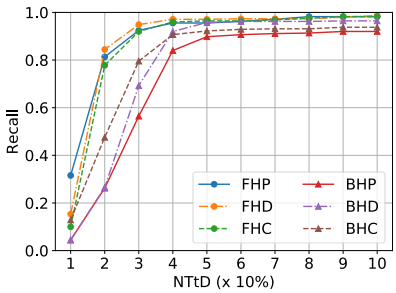


(a) Precision at 100Hz

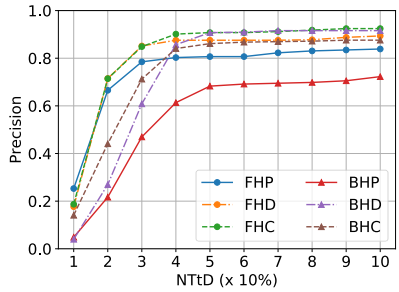
(b) Recall at 100Hz



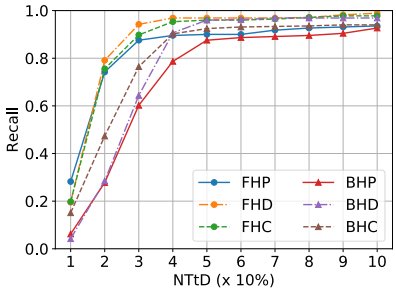
(c) Precision at 50Hz



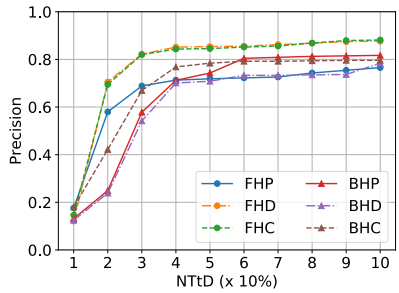
(d) Recall at 50Hz



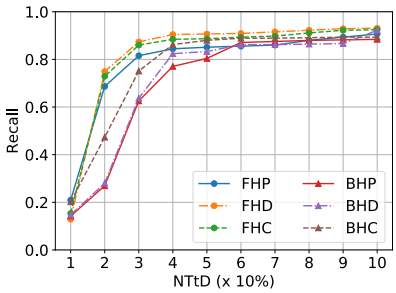
(e) Precision at 25Hz



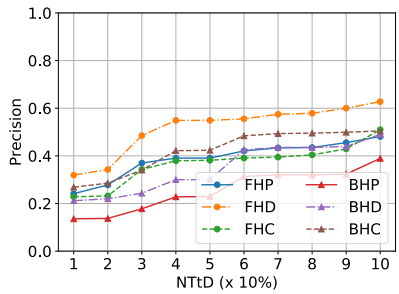
(f) Recall at 25Hz



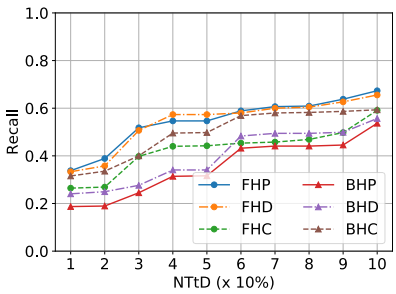
(g) Precision at 12.5Hz



(h) Recall at 12.5Hz



(i) Precision at 6.25Hz



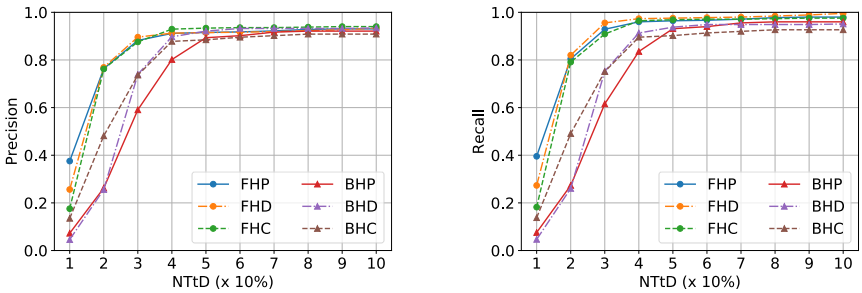
(j) Recall at 6.25Hz

Figure 5.5: Early gesture recognition performance at different sampling rates.

icantly affected even if the sampling rate drops down to 25Hz. But at 25Hz, the *backhand push* gesture has much lower precision compared to other gestures, but then BHP class re-gains a higher precision when the sampling rate drops down to 12Hz. Possibly, during BHP session, there were some movements around 6Hz when the participants were not playing. However, a common trend is that the performance (both precision and recall) drops slowly until 12.5Hz, with the precision and recall then dropping significantly (a 'knee in the curve') when the sampling rate is 6.25Hz. It is possible that the significant signal frequency (for Table Tennis gestures) is above 6.25Hz, so sampling at lower frequencies effectively cause loss of crucial gestural signals and features.

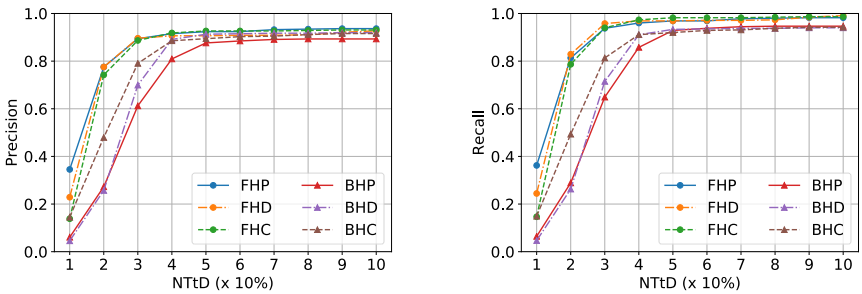
Early recognition with different number of HMM states

The default version of the gesture recognition uses 10 states for each HMM sub-model (e.g, each gesture). It is possibly that the performance of the gesture recognition pipeline is affected when the number of HMM states changes as the complexity of the model changes. Figure 5.6 shows the gesture recognition accuracy with different number of states. When the number of states increases to 15 and 20, the accuracy does not change noticeably. However, when the number of states decreases down to 5, the early recognition ability is significantly affected. Though the final recognition accuracy is not significantly decreased, the accuracy at 40% of gestures is much lower.



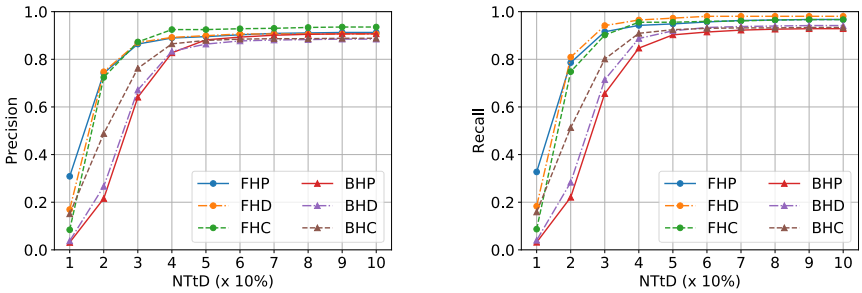
(a) Precision with 20 states

(b) Recall with 20 states



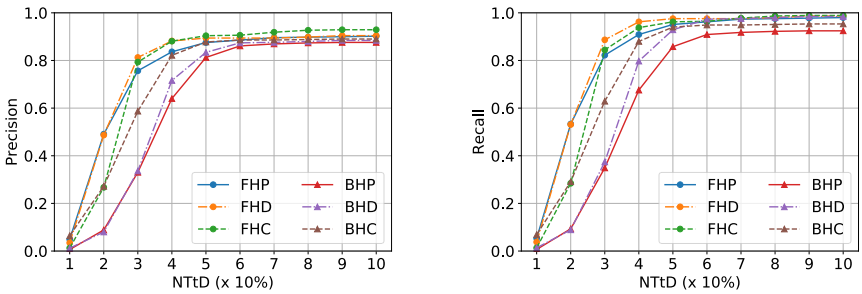
(c) Precision with 15 states

(d) Recall with 15 states



(e) Precision with 10 states

(f) Recall with 10 states



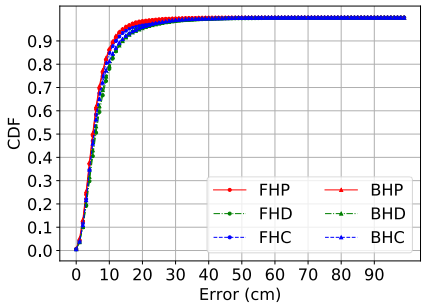
(g) Precision with 5 states

(h) Recall with 5 states

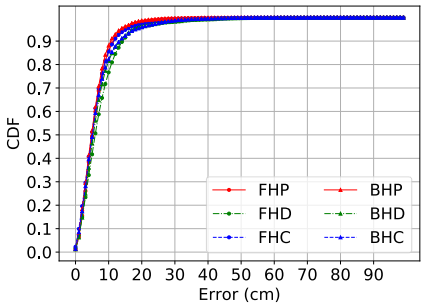
Figure 5.6: Early recognition performance with different number of HMM states.

5.1.2 Hand tracking

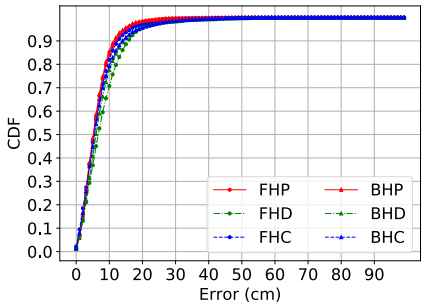
Hand tracking under different sampling rates



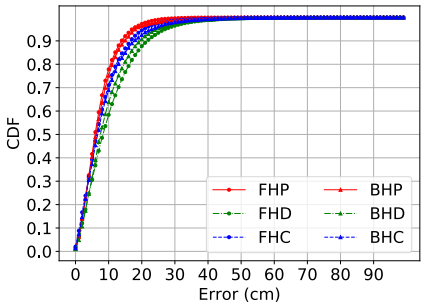
(a) Tracking error at 100Hz



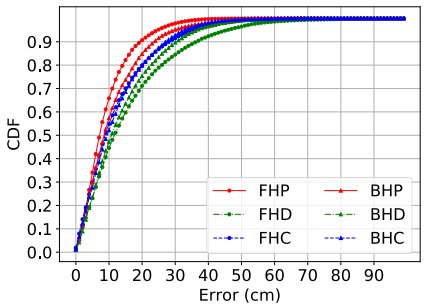
(b) Tracking error at 50Hz



(c) Tracking error at 25Hz



(d) Tracking error at 12.5Hz



(e) Tracking error at 6.25Hz

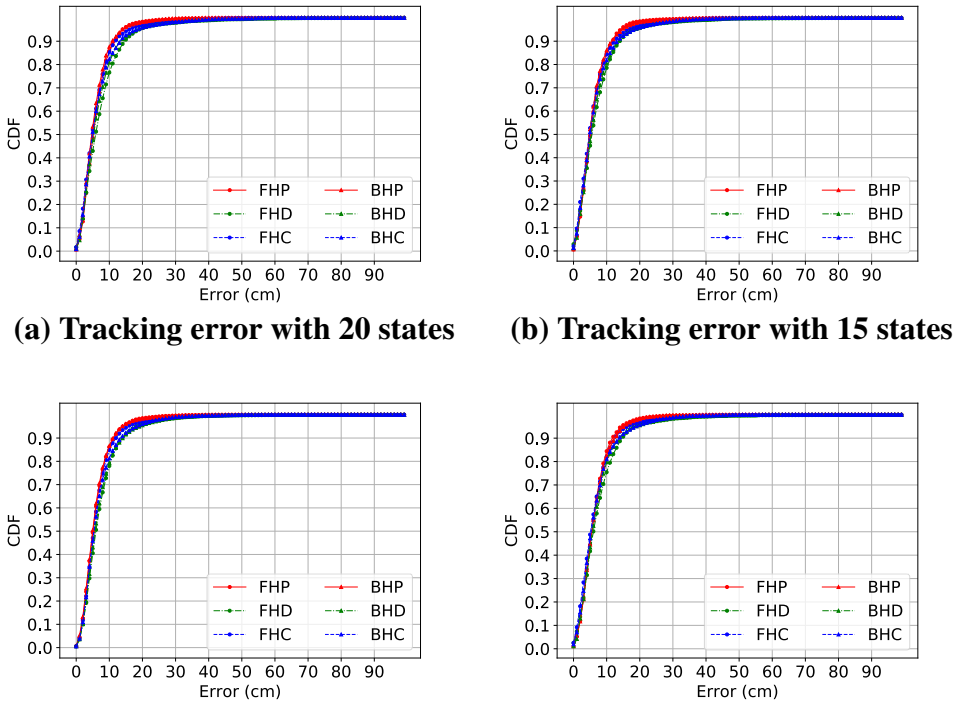
Figure 5.7: Hand tracking performance at different sampling rates.

Similar to the case of early gesture recognition, the tracking accuracy degrades slowly until the sampling rate is 12.5Hz, with a median error of nearly 8cm, then it suddenly increases to around 10cm when the sampling rate is 6.25Hz. At this sampling rate, as the gesture state is not accurate (as shown in Figure 5.5), the gesture recognizer actually does not help to improve the tracking accuracy.

These analyses suggest that our early gesture recognition and tracking can

be used with much lower sensor sampling rate and still achieve comparably high accuracy. This is extremely important for the technique to be deployed on ultra low-power devices as sensors drain more energy at higher sampling rate. In the next sections, I shall investigate if our method can be applied to a *WiWear* device at some useful sampling rates.

Hand tracking with different number of HMM gestures



(a) Tracking error with 20 states **(b) Tracking error with 15 states**

(c) Tracking error with 10 states **(d) Tracking error with 5 states**

Figure 5.8: Hand tracking performance with different number of HMM states.

Though the HMM is used in the gesture recognition pipeline only, the state values are used as features for the hand tracking model (MLP). So it is possible that the tracking error is also affected by the number of states in the HMM model. Figure 5.8 shows the tracking errors with different number of the HMM states. As expected, the median error with 15 and 20 states is not considerably decreased (from 6.2cm compared with 6.0cm). Interestingly, the error with 5 states is not significantly lower either (6.2cm compared with 6.5 cm). As described in Chapter 4, the hand tracking model uses average state values as features (not

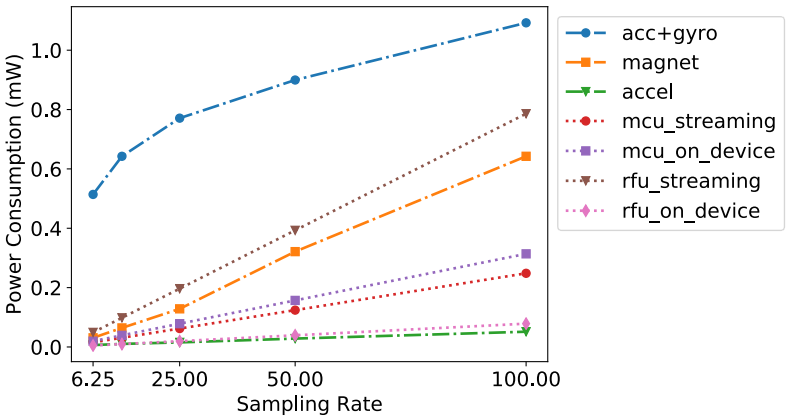


Figure 5.9: Power consumption of different components at different sampling rates (Hz)

the discrete state values), so it is less affected by the number of states because the state values can still indicate the full progress of gestures (e.g, 3.5/5.0 equals to 7.0/10.0). However, the system cannot detect the gesture early, and it is confused that the current estimated position belongs to a *Null* gesture and ignores it.

5.2 Effects of Varying Sampling Rates and Model Complexity on System Power Consumption

Before analysing any sensing technique on *WiWear* wearable, I shall show the energy consumption of hardware components used in a *WiWear* wearable based on the data-sheets of manufacturers. Figure 5.9 shows energy consumption of 3 sensors (accelerometer, gyroscope and magnetometer) from ST [11, 9, 10], the micro-controller (MCU) also from ST [14] and the rf-transceiver (RFU) from Nordic [12]. The energy consumption values at sampling rates that are not directly stated in the data-sheets are interpolated or extrapolated. Though the sampling rate does not directly modify the MCU or the RFU, it has an implicit impact on their power consumption as well. When a sensor is active, it will signal the MCU to wake-up and read the data. As I target real-time applications, I assume that the MCU needs to read the data as soon as the data is available

(equal to the sensor sampling rate). If the MCU needs to transmit the data back to an AP, it signals the RFU to wake-up and transmit the data using RF signal. Accordingly, the data rates of sensor reading and RF transmission affect the active time of the MCU and RFU. Currently I use a data rate of 2Mbps. Another point, that also affects the energy consumption, is the deployment mode. There are two possible, commonplace approaches to execute early gesture recognition & tracking with *WiWear* wearables.

Streaming data mode: In this case, the wearable transmits all the sensor data to the host device (*WiWear* AP) as soon as possible (after every sample), and the actual gesture recognition/tracking algorithms are deployed and executed on the host device. But note that data transmission at high rate with short packets is extremely costly, and this approach effectively saves computational energy but can incur significantly higher communication energy overheads.

On-device processing mode: Alternatively, the *WiWear* wearable executes the algorithm on its local processor if the algorithm is low-complexity enough, and transmits only the results to the AP. This approach, on the contrary, tries to save the expensive communication energy, but sacrifice the MCU power to process the data.

However, an ultra low-power MCU usually has lower power consumption compared to an RFU, and the optimal solution might be completely different depending on how long the MCU processes the data as well as how much data the RFU needs to transmit. In the Figure 5.9, we can see that the power consumption of the MCU using **streaming data** is a bit lower than **on-device** processing. However, the power consumption of the RFU is much higher in the streaming mode compared to on-device processing mode. Depending on how long the processing time is, processing the data on the device (on-device processing) might outperform streaming the data to an AP to be processed at the AP or vice versa.

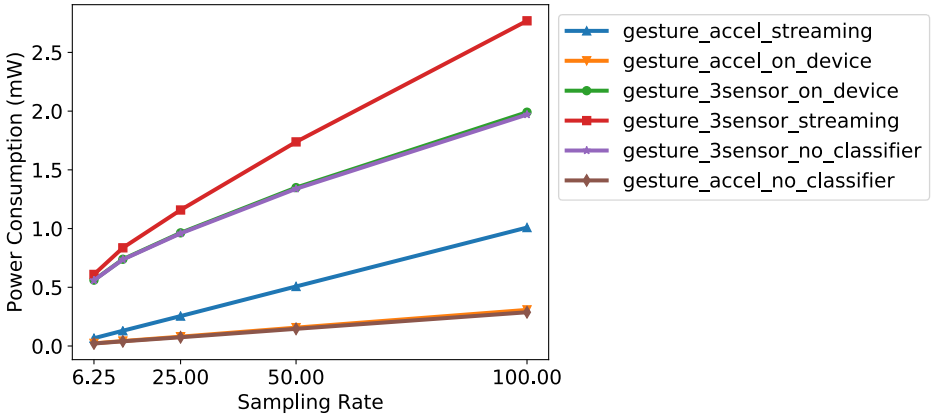


Figure 5.10: Power consumption of gesture recognition at different sampling rates.

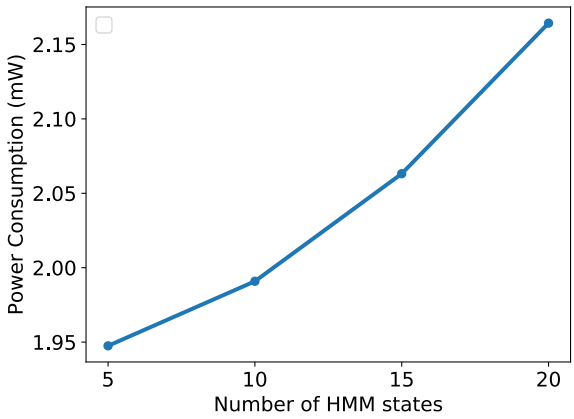


Figure 5.11: Power consumption of gesture recognition with different number of HMM states (sampling rate = 100 Hz).

5.2.1 Energy consumption of gesture recognition on *WiWear* wearable

Currently, due to the lack of support of machine learning models on ultra low-power micro-controllers, I shall project the complexity of machine learning models used in the implementation to the micro-processor processing capabilities, and compute the estimated energy consumption of the algorithm. This projection also enables me to compare the performance of my techniques with a recent work on real-time hand tracking using inertial sensors [83].

As described in Chapter 4, the entire system consists of two main components: (1) gesture recognition model, and (2) hand tracking model. The gesture recognition model consists of a Hidden Markov Model (HMM) and a Random Forest model. The HMM computes raw probabilities that each sample belongs

to a specific states. The Random Forest uses these state probabilities values as part of a feature vector to determine if a sample actually belongs to a gesture or not. The hand tracking model is a Multi-Layer-Perceptron regression model. It also uses the state probabilities values as part of a feature vector to estimate the hand position. Note that the hand tracking model always need the HMM to estimate the raw state probabilities. In total, our algorithm uses 5 machine learning models: (a) an HMM and, (b) a Random Forest model to recognize gesture early; (c) 3 Multi-Layer-Perceptron (MLP–a simple neural net) to estimate hand position (3 axes). The computational cost of each of these 5 models is computed as follows:

- The HMM includes 6 10-state sub-models (each state in a sub-model can transit to only states in that sub-model) for 6 gestures. So the probabilities update takes 600 operations. The input data is quantized into one of 256 values, so it takes 256 operations for quantization. The emission probabilities are implemented as a look-up table, so it takes only one look-up. So in total, the HMM model needs 856 operations.

- The number of trees in the Random Forest is set at 10. The input is a vector of 21 values, and the max depth of trees is set at 21. In the worst case, it needs to spend 210 operations to search through all trees (each tree has a depth of 21–worst case) for a prediction.

- The MLP includes 3 hidden layers with the size of (9, 18, 9), and the input size of 15. Layers in MLP are fully connected. The output has a size of only 1 (x/y/z value). The 3 MLP models need 1404 operations to estimate one sample.

In total, the number of operations is 2470 operations. Usually, each operation includes a multiplication and an addition. So the total number of CPU instructions is 4940. The Miro-controller CPU frequency is 16MHz, and it supports both addition and multiplication. So the total time needed for a recognition and

estimation is 0.31ms. Depending on the sampling rate of the sensors, the total active time of the micro-controller will be different, and the active time is computed from this sampling rate. For example, if the sampling rate is 100Hz, the active time of the MCU to process data in 1 second is 31ms ($0.31ms \times 100$), and thus the average power is 0.2325mW (7.5×0.031). Figure 5.10 shows the power consumption of only gesture recognition component in 2 settings (a) all 3 sensors (acclerometer, gyrscope and magnetometer), and (b) just the accelerometer sensor alone. In each setting, the power consumption of the full gesture recognition pipeline (HMM + Random Forest) or only the HMM (excluding the Random Forest classifier) is also shown. For this candidate gesture recognition application, we see that the total power consumption is affected significantly by (1) the use of additional energy-hungry sensors (gyroscope, magnetometer), and (2) frequent RFU usage (as consequence of which streaming mode is typically more power-hungry than on-device computation). The inclusion/exclusion of the Random Forest classifier causes inconsiderable effect on the total energy consumption as the prrediction time of the Random Forest is much less than the time the MCU spends on reading sensors and transmitting data.

As the number of HMM states contributes the most to the computation of the gesture recognition. The change in the number of states will affect the energy consumption of the system. Figure 5.11 shows the power consumption of the a *WiWear* wearable if the gesture recognition algorithm is executed on the wearable using all 3 sensors (accelerometer, gyroscope and magnetometer) with a sampling rate of 100Hz. In this case, the streaming energy consumption is not shown because in the case of streaming data, the wearable does not execute the gesture recognition task. Figure 5.11 shows a noticeable increase in energy consumption. Increasing the number of states from 5 to 10 incurs $50\mu W$. Different applications might perceive these relative power differences to be significant (or not). Moreover, depending on the exact latency bound that an application must meet, it can additionally determine if 10 or 5 states is more suitable.

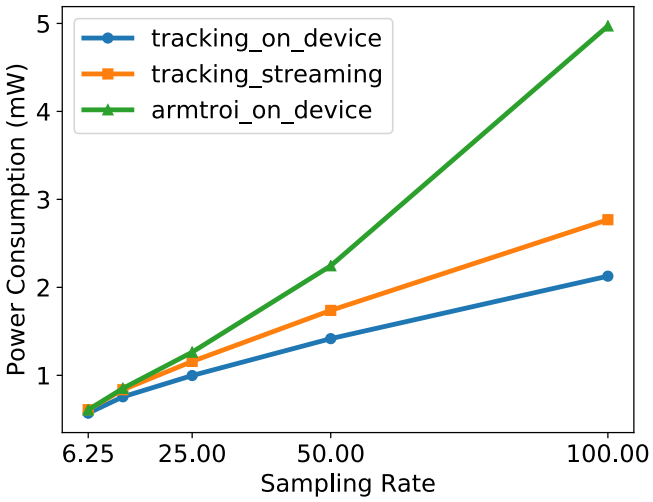


Figure 5.12: Power consumption of hand tracking at different sampling rates.

5.2.2 Energy consumption of hand tracking on *WiWear* wearable

In the previous section, I showed the power consumption of only gesture recognition on a *WiWear* device. I now study the corresponding power consumption profile for the additional "hand tracking" functionality. As our hand tracking algorithm utilizes the gesture as an input, its power profile implicitly includes the power consumed by gesture recognition. Figure 5.12 shows that the power consumption using on-device processing mode is still lower than the streaming mode, but the difference is lower than the case of gesture recognition. The reason is that the MCU needs to spend more time to estimate the wrist position.

To position our technique among related studies on arm tracking, relative comparisons are provided in Table 5.2. The NFC ring can recognize finger gestures with lower power consumption compared to our system, but it does not support arm tracking. The system proposed by Wittmann et al. is potentially a low-latency and low-power arm tracking technique for battery-less devices as it requires only several multiplications to estimate wrist position using arm orientation and arm length. However, it requires several sensors to be worn at different body parts (chest, lower arm, upper arm). ArmTrak supports arm tracking using only one smartwatch, but its processing time is extremely high.

ArmTroi addressed the high processing time of ArmTrak by applying a hierarchical search technique. Like our system, ArmTroi target is to support real-time arm tracking and gesture/activity recognition. The reported power consumption of ArmTroi was measured on a smart phone. So it is interesting to see the projected power consumption of ArmTroi on the same micro-controller used in our system (only the arm tracking function). Figure 5.12 shows that ArmTroi consumes more energy than our technique, and the difference is even higher at higher sampling rates.

Table 5.2: Comparison of recent motion sensing studies.

Study	**Accuracy**	**Tracking error**	**Processing time**	**Power consumption**	**# Devices**
NFC Ring [55]	70% (12 gestures)	NA	250ms (1MHz micro-controller)	$731\mu W$ (1MHz micro-controller)	1
Wittmann [130]	NA	6° (arbitrary movement)	61ms (end-to-end, including transmission time and screen response)	Low	3
ArmTrak [113]	NA	9.2cm (arbitrary movement)	10s (1s segment, desktop)	High	1
ArmTroi [83]	92% (17 activities)	11.5cm (arbitrary movement)	150ms (1s segment, desktop)	1776mW (Phone)	1
Our technique	93% (6 gestures)	6.3cm (during 6 gestures)	1.4ms (Android watch)	1.95mW (projected, 16MHz micro-controller)	1

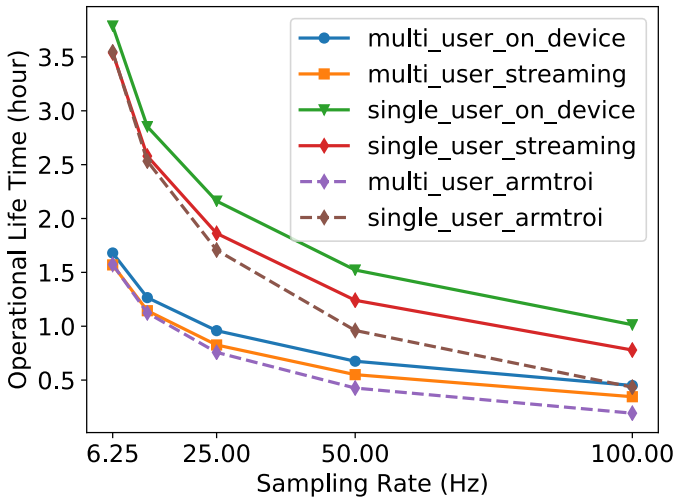


Figure 5.13: Operational life time of *WiWear* wearable, in single-user and multi-user scenarios, at different sampling rates.

5.2.3 Operational life time of gesture recognition & tracking on *WiWear* wearable

In this section, I shall compare how long and how fine-grained the *WiWear* wearable can support continuous real-time sensing. I shall also compare our technique with a recent hand tracking technique, ArmTroi [83]. ArmTrak [113] proposed a hand tracking algorithm using inertial sensors, but its high complexity makes it unlikely to be used in real-time applications. The complexity comes from the large state space of the HMM model. This method relies on a known upper arm length of a user and tries to estimate the user elbow location in a sphere with a radius of the upper arm length. So each possible location is modeled as a HMM state. To increase the robustness of state estimation, each HMM state is actually modeled as a pair of consecutive elbow locations. To determine the most probable elbow locations, it needs to observe a segment of sensor readings and uses Viterbi algorithm to find out the most probable sequence of states corresponding to the segment of sensor readings. Consequently, the complexity is O(N^3T), where N is the number of possible elbow locations, T is the number of samples in a sensor segment. Assume that the elbow can move freely in part of a sphere with a radius (upper arm length) of 25cm, and a coverage angle of 90° (a hemisphere). This probable space has an area of $3927cm^2$ ($2\pi 25^2$). If

this space is divided into $5cm \times 5cm$ squares, there will be 157 elbow positions. Assume that the algorithm computes the elbow positions for every 1-second sensor segment (T = 100 at 100Hz), it needs 387 millions operations. ArmTroi [83] addresses this issue by limiting the search space from a state to only the most probable ones, and also apply hierarchical search to reduce the complexity to $O((\frac{N}{n1})^2 T)$, where n is the first grid division with the default value of 10. So the total number of operations to compute a 100-sample trajectory is reduced to 24649, and thus the total needed instructions in a micro-controller is 49298.

There are several assumptions in this analysis. I assume that the device can harvest an average power of $90\mu W$ throughout a day (in single user case), which is validated in our experiment with our prototype. As the harvested power of the prototype is almost one third of the simulated value in the same condition, I assume that the efficiency of a real prototype is 30% of that achieved under ideal conditions. Accordingly, for a multi-device scenario, the harvested power is $\sim 40\mu W$. I also assume that a user performs one gesture in every 10 seconds, and the wearable transmits estimated hand position corresponding to each sample in a 1-second window to the AP whenever a gesture is done (on-device processing method). For the ArmTroi approach, it is not clear how the duration of the "tracking windows" are defined–i.e., whether it tracks overlapping consecutive windows (which requires running the algorithm for every sample) or uses segmented 'tumbling' windows (which will decrease the accuracy for several initial samples). Accordingly, I assume that it runs with continuous windows (1 sample shift) so as to achieve the highest possible tracking accuracy. In addition, my estimation of this alternative ArmTroi technique is conservative, as I do not consider the energy overhead of additional operations, such as the Gaussian calculation for transition probabilities (compared to our Look-up table). The Gesture recognition model (based on estimated position) is not considered either, as there is not enough information to infer the number of operations of that deep learning model. It is also likely that the model is too

large for an ultra low-power wearable. Figure 5.13 plots the total *operational lifetime* (defined as the maximum daily duration for which such tracking can be sustained such that the total energy drain equals the daily harvested energy) of our technique (which always includes an initial gesture recognition step) on a *WiWear* wearable, compared to that of ArmTroi. Both are quite similar at extremely low sampling rate (6.25Hz), but ArmTroi's active operational lifetime decreases quickly when sampling rate increases. Figure 5.13 shows that, even in multi-device (4 devices or users) scenario, at the highest sampling rate of 100Hz, the device can support continuous real-time sensing, using our technique, for 0.34 hour if streaming data back to AP, but can extend the operational lifetime to 0.44 hour if running the algorithm on-device. In the same condition, ArmTroi can support only 0.19 hour of continuous sensing. At a sampling rate of 25Hz, our hand tracking accuracy just slightly decreases, but the operation time can be extended up to 0.95 hour and 0.82 hour using on-device processing and streaming method correspondingly (in contrast to 0.75 hours for ArmTroi).

5.3 Discussion

Previously in this Chapter, I have investigated the performance of our early gesture recognition & tracking to under different sampling rates which is a key factor to reduce energy consumption of sensors. Though I have not deployed the algorithm on the ultra low-power *WiWear* device due to the lack of library support, the complexity analysis of each component in the algorithm suggests that it is possible to deploy such an low-complexity algorithm on a *WiWear* wearable. In particular, using a relative low sampling frequency of 25Hz, allows us to achieve early gesture recognition (with F1-score of 90%) and operate a single WiWear device for a period of 2.16 hours/day, with the gesture recognition logic being performed directly on the wearable platform. Based on the assumption that each operation in the gesture recognition algorithm can

be mapped 1-to-1 to the micro-controller instruction, the on-device processing method outperforms the streaming method. But there could be cases that one operation in the algorithm is translated into several CPU instructions. In that case, the streaming might outperform the on-device processing method. However, in both situations, my work suggests that it is possible to deploy early gesture recognition & tracking techniques on *WiWear* wearables, and thereby support the new class of real-time, interactive gesture-based applications that I target.

Chapter 6

Conclusion and Future Directions

This dissertation has shown that it is possible to enable real-time motion sensing on energy-harvesting wearables by incorporating various techniques to overcome (i) the limitations of energy harvesting for wearables in indoor environments, and (ii) high complexity & latency of gesture recognition and tracking using wearable inertial sensors. In this chapter, I shall summarize the key contributions of this dissertation and discuss the future directions to be explored.

6.1 Summary of Contribution

In this dissertation, I have proposed a new way of powering wearable devices through WiFi-based beamforming, and a real-time low-complexity gesture recognition and tracking which can be applied to energy-harvesting devices.

6.1.1 WiWear: Battery-less Motion Sensing

In chapter 3, I described *WiWear*, an approach for battery-less motion sensing that uses beam-formed WiFi transmissions to deliver energy to energy-harvesting wearables. *WiWear* platform can co-exist with, and even augment, other energy harvesting technologies (e.g. light, motion) to enable energy-hungry real-time motion sensing.

I also proposed an algorithm to optimize the antenna power pattern to support multiple wearables concurrently moving around the AP. The numerical analysis results suggest that one *WiWear* AP can support multiple devices. In a range from 1m to 3m, it can support 4 devices even with real-time sensing thank to the beam shaping algorithm.

To validated the viability of the system in realistic conditions, I have implemented one prototype of the *WiWear* AP and one prototype of the *WiWear* wearable. The AP supports complete functions of an 802.11g AP, and supports AoA estimation & beam-forming to detect the direction of the wearable and beam-form energy toward that direction. The wearable is able to harvest energy from WiFi signal, and apply smart event-based operations using zero-energy motion detector to operate a full accelerometer-equipped wearable device.

To evaluate the capability of the system to deliver energy to the wearable via WiFi transmissions (UDP packets). I have conducted several experiment in an office meeting room environment. The wearable receiver can harvest more than 400μW of energy at a distance of 1 meter, and more than 30μW at 3 meters. Though lower than the simulated values, this is expected as the imperfection of devices and environment, and the wearable is still able to use the harvested energy to record accelerometer data and transfer it back to the AP.

I have also conducted a constrained user study to evaluate the viability of the system when the wearable is worn by a user in a representative office setting. *WiWear* wearable achieve positive net energy (after spending on recording accelerometer and transfer data in burst mode) in all 4 positions (4 participants) with a distance from 1.2m to 2m. These results have suggested that *WiWear* can be a viable approach for battery-less motion sensing such as gestures/activities recognition.

6.1.2 Early Gesture Recognition & Tracking

In Chapter 4, I described a novel early gesture recognition & tracking technique which can recognize gestures early, even before the gesture has finished, and track the position of user's hand with low latency. Being able to track gestures with low-latency is crucial for interactive applications, and the low complexity of the algorithm make it applicable to energy-harvesting wearables.

To enable early gesture recognition, I have identified the problem of existing gesture recognition techniques which is the requirement of segmentation to be done before actual classification of the segment into different gestures. It implies that a system has to wait until the end of a gesture, which is too late when applied to interactive applications. I have shown that by smartly construct a time series processing model (in this case, an HMM), the segmentation can be omitted, and thus enable early gesture recognition.

Using a representative application of Virtual Table Tennis, I described a novel method which can recognize Table Tennis gestures before the gesture ends. Then I explained the how low-complexity (and thus low latency) hand tracking can be realized based on the gesture being recognized. Throughout the experiments with 10 novice Table Tennis players and 15 experienced players, our technique achieved more than 92% recognition accuracy within the first 50% of the gestures, and a tracking median error less than 6.5cm. I believe that the proposed technique is applicable to many other interactive applications such as virtual sport coach (e.g. for volley ball) or virtual rehabilitation assistant.

To achieve low-complexity hand tracking, I have presented a gesture-based tracking approach which utilize the information the progress of the gesture being recognized and other supported IMU sensor features to estimate user's hand position. This is based on an observation that each type of gestures usually have different trajectories, and gestures of the same type usually have similar (but not identical) trajectories. As the gesture recognition helps narrow down the probable space of hand position, a simple regression model (in this case, a

3-layer neural network) is able to estimate hand position accurately.

The entire system of gesture recognition and tracking is low-complexity and able to recognize gestures early. The entire system was deployed on a Samsung Gear Live smartwatch. Through a user study with 5 participants with the task of hitting a real, suspended Table Tennis ball, 85.7% of times, participants feel the audio feed-back of the smartwatch (when a "hit" is estimated) is at the same time as the physical "hit" moment. These results showed that a low-complexity early gesture recognition and tracking was, indeed, achieved.

6.1.3 Feasibility Analyses of Early Gesture Recognition & Tracking on Energy-harvesting Wearable

In Chapter 5, I investigate the feasibility of early gesture recognition and tracking on a *WiWear* wearable which harvests energy from WiFi signals. I first studied the robustness of our early gesture recognition and tracking technique under different sampling rates. The results showed that the performance of the system only degrades slightly when sampling rate drops from 100Hz (original) down to 25Hz. With these sampling rates, I have analysed the viability of a *WiWear* wearable when it executes early gesture recognition and tracking algorithm. The analyses suggest that the *WiWear* device can support early gesture recognition and tracking for 2.16 hours of active running with a continuous average harvested power of $90\mu W$ (throughout a day) under the sampling rate of 25Hz which results in 90% accuracy. Even in a multi-user scenario (4 users), the system can still support gesture recognition and tracking for almost 1 hour. The analyses also show that our gesture recognition and tracking technique outperforms ArmTroi [83] in operational life time if deployed on a *WiWear* wearable.

6.2 Reflections and Lessons Learned

Throughout the studies of the two key research pieces in this dissertation, I have learned two valuable lessons that I believe are important for these types of studies.

- **Need for identifying the right target user segment early:** For studies whose main material comes from user studies, having a right target for the participants at the beginning is crucial. An experiment requires a great amount of time and effort. To recruit participants for an experiment is hard, to recruit appropriate participants is even harder, especially, if we need participants with some special skills (e.g, play Table Tennis). If we recruit wrong target participants, we need to spend time and effort to conduct the experiment, but later the experiment results are insignificant (or even invalid). Having a clear, detailed selection criteria (e.g, 3-year experience, regular player) at the beginning is crucial to avoid this type of problems.

- **Be aware of signal leakage even when the transmitter is inactive**: In multi-antenna device such as our multi-antenna AP, the antennas are very close to each other. Because of the imperfection of circuit inside transceiver ICs, the transmitter still transmits a weak continuous cosine wave even if it is inactive. It would not be a problem if the receiving antenna is far away. But if the antennas are relatively close to each other, this will be added to the receiving antenna. Though this leakage is weak, it still causes problems if the receiving antenna increases it gain to listen to signal afar. To avoid this, switch the antenna to inactive receiving mode instead of inactive transmitting mode.

6.3 Discussion & Future Direction

Through this dissertation, I have shown that it is possible to enable energy-hungry motion sensing on battery-less wearables using energy harvested from smartly beam-formed WiFi signals. I have also shown that a low-complexity, low-latency gesture recognition and tracking using inertial sensors was achieved. The numerical analyses suggested that low-latency regesture recognition & tracking is feasible on energy-harvesting wearables. However, this dissertation still has several limitations. Though *WiWear* prototype and the numerical analyses partially proved the feasibility of the *WiWear* vision, additional, longer-term, in-the-field deployments are needed before the viability of the WiWear vision is conclusively demonstrated. The current prototype has also intrinsic limitations of third-party components which reduce the efficiency of power delivery. At present, my techniques make it possible to support robust early gesture detection (using just low sampling-rate accelerometer data), but is insufficient to support highly accurate hand tracking (as this requires a more energy-hungry gyroscope sensor) for long active using sessions. The lack of support for machine learning libraries on pow-power embedded micro-processors also hinders the implementation of early gesture recognition & tracking algorithm on the *WiWear* prototype. However, these limitations are addressable.

This dissertation has also laid a basis for exploration of further potentials of battery-less motion sensing. Below, I shall discuss several possible future directions.

6.3.1 Extended Capabilities for Battery-less Real-time Motion Sensing

To support extended periods of real-time motion sensing on energy-harvesting wearables, of course, it is unrealistic to assume that sensing algorithms can be

infinitely shrunk to fit in a energy-harvesting device. Another approach would be using an energy harvester to perform both functions: (1) harvest energy, and (2) recognize gesture and/or track hand. For example, using different receiver antennas to receive more energy and also to detect hand orientation (e.g, using phase shift of different antennas). Such an idea has been explored [78], for example, in using the temporal patterns of light harvesting by photodiodes attached to a smartwatch/smartglasses to determine the touch gestures performed. Especially, upcoming millimeter wave technologies, which provide fine-grained sub-cm spatial resolution, will definitely unleash fine-grained orientation sensing to a new level. Or similar to the motion trigger (described in Chapter 3) which can detect significant hand movement and generate some energy. We can also think of more futuristic devices, such as a piezoelectric patch, that intrinsically perform both energy harvesting and sensing. The deformation of our wrist could generates energy through the piezoelectric patch and also provide some information of muscle contraction, which can be used to recognize gestures.

6.3.2 Smart Multi-AP Scheduling for WiFi-based Energy Harvesting Wearables

In our *WiWear* vision there can be multiple APs to serve multiple wearables. Similarly, in practice, many APs and repeaters could be densely deployed in indoor environment (e.g, in a foodcourt). Scheduling which AP to serve which energy-harvesting device would provide a two-fold benefit: (1) reduce the number of devices one AP has to serve, and thus increase the energy density at other devices, and (2) balance the energy versus data communication load. Also, multiple APs can enable accurate device localization (e.g, using triangulation) which can be exploited for better scheduling, and many other applications. In addition, the presence of multiple WiFi APs leads to their possible coordinated function–techniques such as Energy-Ball [51] have already provided early demonstrations of such possibilities, albeit under unrealistically dense in-

frastructure deployment. With the increasing trend of cloudRANs [43], which provide centralized, cloud-based control of AP infrastructures, such coordinated control is likely to become increasingly feasible.

6.3.3 Enhanced Battery-less Hand Tracking using WiFi Signal

In this dissertation, hand tracking using IMU sensors was studied which can be applied on battery-less wearables. But hand tracking without IMU might be achieved without IMU sensors using multiple APs, or using the CSI information from the AP to enhance the hand tracking using less energy-hungry sensors (e.g. using only accelerometer and magnetometer without using gyroscope). For example, if there are multiple APs (e.g, 3 APs) installed near a big screen for playing games, the APs can collectively estimate micro position of user's hand with a battery-less device on it by tracking the periodic "ping" packets from the device. In a less extreme case, the application in a AP can combine the CSI information from the device with the accelerometer and magnetometer reading from the device to accurately estimate hand position. This might be possible as it is known that the movement will create Doppler effect which is captured by the receiving antenna.

Bibliography

[1] Best heart rate monitor 2019: Hrm watches and chest straps compared. https://www.wareable.com/fitness-trackers/best-heart-rate-monitor-and-watches. Accessed: 2019-05-29.

[2] History of the only manufacture with every watchmaking expertise. https://www.seikowatches.com/global-en/special/heritage/, 2012. Accessed: 2019-10-05.

[3] Ultra-low-power arm cortex-m0+ mcu with 64 kbytes flash, 32 mhz cpu, usb, lcd. https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/detail.cfm?mdrfoi__id=2913825, 2012. Accessed: 2019-10-05.

[4] Wearable technology market growing at a cagr of 15.5% and expected to reach $51.6 billion by 2022 - exclusive report by. https://www.bloomberg.com/press-releases/2019-07-01/wearable-technology-market-growing-at-a-cagr-of-15-5\-and-expected-to-reach-51-6-billion-by-2022-\exclusive-report-by, 2012. Accessed: 2019-10-05.

[5] Warp wireless open-access research platform. https://mangocomm.com/products/kits/warp-v3-kit, 2017. Accessed: 2017-08-25.

[6] 802.11 reference design for warp v3. https://warpproject.org/trac/wiki/802.11, 2018. Accessed: 2018-09-29.

[7] Amk432bj477. https://octopart.com/amk432bj477mm-t-taiyo+yuden-61790332#, 2018. Accessed: 2018-04-08.

[8] Fm25vn10g. https://mw.infinite-electronic.hk/pdf/d7e4771220/FM25VN10-G.pdf, 2018. Accessed: 2018-04-08.

[9] Iis2mdc. https://www.st.com/en/mems-and-sensors/iis2mdc.html, 2018. Accessed: 2018-04-08.

[10] Ism330dlc. https://www.st.com/en/mems-and-sensors/ism330dlc.html, 2018. Accessed: 2018-04-08.

[11] Lis3dh. https://www.st.com/en/mems-and-sensors/lis3dh.html, 2018. Accessed: 2018-04-08.

[12] nrf24 series. https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF24-series, 2018. Accessed: 2018-04-08.

[13] Solepower smart boots. http://www.solepowertech.com/smartboots/, 2018. Accessed: 2018-04-04.

[14] Ultra-low-power arm cortex-m0+ mcu with 64 kbytes flash, 32 mhz cpu, usb, lcd. http://www.st.com/en/microcontrollers/stm32l053r8.html, 2018. Accessed: 2018-04-08.

[15] Focals smart glasses: Buy them anywhere, wear them everywhere. https://www.bynorth.com/, 2019. Accessed: 2019-10-10.

[16] Google home assistant. https://developers.google.com/assistant/smarthome/faq, 2019. Accessed: 2019-10-10.

[17] Heart guide. https://omronhealthcare.com/products/heartguide-wearable-blood-pressure-monitor-bp8000m/, 2019. Accessed: 2019-10-10.

[18] Hololens 2. https://www.microsoft.com/en-us/hololens, 2019. Accessed: 2019-10-10.

[19] Maxim ecg monitor. https://www.maximintegrated.com/en/products/sensors/MAX-ECG-MONITOR.html, 2019. Accessed: 2019-10-10.

[20] Occulus. https://www.oculus.com/?locale=en_US, 2019. Accessed: 2019-10-10.

[21] Panasonic amorphous silicon solar cells. https://panasonic.co.jp/ls/psam/en/products/pdf/Catalog_Amorton_ENG.pdf, 2019. Accessed: 2019-10-10.

[22] T02 heathy smart band body temperature blood oxygen heart rate monitor. https://gearvita.com/t02-heathy-smart-band-body-temperature-blood-oxygen-\heart-rate-monitor.html, 2019. Accessed: 2019-10-10.

[23] H. Abdelnasser, M. Youssef, and K. A. Harras. Wigest: A ubiquitous wifi-based gesture recognition system. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1472–1480. IEEE, 2015.

[24] S. Agrawal, I. Constandache, S. Gaonkar, R. Roy Choudhury, K. Caves, and F. DeRuyter. Using mobile phones to write in air. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 15–28. ACM, 2011.

[25] O. Amft, H. Junker, and G. Troster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 160–163. IEEE, 2005.

[26] W. T. Ang, P. K. Khosla, and C. N. Riviere. Kalman filtering for real-time orientation tracking of handheld microsurgical instrument. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2574–2580. IEEE, 2004.

[27] E. K. Antonsson and R. W. Mann. The frequency content of gait. *Journal of biomechanics*, 18(1):39–47, 1985.

[28] M. Bächlin, K. Förster, and G. Tröster. Swimmaster: a wearable assistant for swimmer. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 215–224. ACM, 2009.

[29] P. Bahl, V. N. Padmanabhan, V. Bahl, and V. Padmanabhan. Radar: An in-building rf-based user location and tracking system. 2000.

[30] R. K. Balan, A. Misra, and Y. Lee. Livelabs: Building an in-situ real-time mobile experimentation testbed. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, page 14. ACM, 2014.

[31] O. Bau and W. E. Mackay. Octopocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 37–46. ACM, 2008.

[32] F. Bevilacqua, B. Zamborlin, A. Sypniewski, N. Schnell, F. Guédy, and N. Rasamimanana. Continuous realtime gesture following and recognition. In *Gesture in embodied communication and human-computer interaction*, pages 73–84. Springer, 2009.

[33] D. Bharadia, E. McMilin, and S. Katti. Full duplex radios. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 375–386. ACM, 2013.

[34] M. Billinghurst and T. Starner. Wearable devices: new ways to manage information. *Computer*, 32(1):57–64, 1999.

[35] P. Blank, J. Hoßbach, D. Schuldhaus, and B. M. Eskofier. Sensor-based stroke detection and stroke type classification in table tennis. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 93–100. ACM, 2015.

[36] P. Blank, T. Kautz, and B. M. Eskofier. Ball impact localization on table tennis rackets using piezo-electric sensors. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, pages 72–79. ACM, 2016.

[37] R. J. Bootsma and P. C. van Wieringen. Timing an attacking forehand drive in table tennis. *Journal of experimental psychology: Human perception and performance*, 16(1):21, 1990.

[38] A. L. Borstad, R. Crawfis, K. Phillips, L. P. Lowes, D. Maung, R. McPherson, A. Siles, L. Worthen-Chaudhari, and L. V. Gauthier. In-home delivery of constraint-induced movement therapy via virtual reality gaming. *Journal of patient-centered research and reviews*, 5(1):6, 2018.

[39] E. Boyer, F. Bevilacqua, F. Phal, and S. Hanneton. Low-cost motion sensing of table tennis players for real time feedback. *Int. J. Table Tennis Sci*, 8:1–4, 2013.

[40] B. Campbell and P. Dutta. An energy-harvesting sensor architecture and toolkit for building monitoring and event detection. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14. ACM, 2014.

[41] B. Campbell, B. Ghena, and P. Dutta. Energy-harvesting thermoelectric sensing for unobtrusive water and appliance metering. In *Proceedings of the 2Nd International Workshop on Energy Neutral Sensing Systems*, ENSsys '14. ACM, 2014.

[42] N. S. Carvalho. *Movement Tracking using Bluetooth Low Energy*. PhD thesis, Trinity College Dublin, 2015.

[43] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann. Cloud ran for mobile networksa technology overview. *IEEE Communications surveys & tutorials*, 17(1):405–426, 2014.

[44] S. J. Chen, T. Kaufmann, and C. Fumeaux. Wearable textile microstrip patch antenna for multiple ism band communications. In *2013 IEEE Antennas and Propagation Society International Symposium (APSURSI)*, July 2013.

[45] S. Cheshire. Latency and the quest for interactivity. In *White paper commissioned by Volpe Welty Asset Management, LLC, for the Synchronous Person-to-Person Interactive Computing Environments Meeting*, 1996.

[46] M. Claypool and K. Claypool. On latency and player actions in online games. 2006.

[47] D. Cook, K. D. Feuz, and N. C. Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and information systems*, 36(3):537–556, 2013.

[48] M. A. El-Gohary. Joint angle tracking with inertial sensors. 2013.

[49] E. Ertin, N. Stohs, S. Kumar, A. Raij, M. al'Absi, and S. Shah. Autosense: Unobtrusively wearable sensor suite for inferring the onset, causality, and consequences of stress in the field. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, pages 274–287. ACM, 2011.

[50] A. Eyck, K. Geerlings, D. Karimova, B. Meerbeek, L. Wang, W. IJsselsteijn, Y. De Kort, M. Roersma, and J. Westerink. Effect of a virtual coach on athletes motivation. In *International Conference on Persuasive Technology*, pages 158–161. Springer, 2006.

[51] X. Fan, H. Ding, S. Li, M. Sanzari, Y. Zhang, W. Trappe, Z. Han, and R. E. Howard. Energy-ball: Wireless power transfer for batteryless internet of things through distributed beamforming. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(2), July 2018.

[52] C. Felini, M. Merenda, and F. G. D. Corte. Dynamic impedance matching network for rf energy harvesting systems. In *2014 IEEE RFID Technology and Applications Conference (RFID-TA)*, Sept 2014.

[53] E. Foxlin. Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter. 1996.

[54] A. Goldsmith. *Wireless communications*. Cambridge university press, 2005.

[55] J. Gummeson, B. Priyantha, and J. Liu. An energy harvesting wearable ring platform for gestureinput on surfaces. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 162–175. ACM, 2014.

[56] A. Hande, T. Polk, W. Walker, and D. Bhatia. Indoor solar energy harvesting for sensor network router nodes. *Microprocess. Microsyst.*, 31(6), Sept. 2007.

[57] J. Hester and J. Sorber. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17. ACM, 2017.

[58] M. Hoai and F. De la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014.

[59] W. Huang, H. Chen, Y. Li, and B. Vucetic. On the performance of multi-antenna wireless-powered communications with energy beamforming. *IEEE Transactions on Vehicular Technology*, 65(3):1801–1808, 2016.

[60] W. A. IJsselsteijn, Y. d. Kort, J. Westerink, M. d. Jager, and R. Bonants. Virtual fitness: stimulating exercise behavior through media technology. *Presence: Teleoperators and Virtual Environments*, 15(6):688–698, 2006.

[61] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti, and P. Sinha. Practical, real-time, full duplex wireless. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 301–312. ACM, 2011.

[62] R. Jota, A. Ng, P. Dietz, and D. Wigdor. How Fast is Fast Enough?: A Study of the Effects of Latency in Direct-touch Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2291–2300. ACM, 2013.

[63] H. Junker, O. Amft, P. Lukowicz, and G. Tröster. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41(6):2010–2024, 2008.

[64] K. Katsuragawa, K. Pietroszek, J. R. Wallace, and E. Lank. Watchpoint: Freehand pointing with a smartwatch in a ubiquitous display environment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 128–135. ACM, 2016.

[65] F. Kawsar, C. Min, A. Mathur, M. Van den Broeck, U. G. Acer, and C. Forlivesi. esense: Earable platform for human sensing. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 541–541. ACM, 2018.

[66] B. Kellogg, V. Talla, and S. Gollakota. Bringing gesture recognition to all devices. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 303–316, 2014.

[67] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *Consumer depth cameras for computer vision*, pages 119–137. Springer, 2013.

[68] M. A. A. H. Khan, N. Roy, and A. Misra. Scaling human activity recognition via deep learning-based domain adaptation. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE, 2018.

[69] D. Kharlamov, K. Pietroszek, and L. Tahai. Ticktockray demo: Smartwatch raycasting for mobile hmds. In *Proceedings of the 2016 Symposium on Spatial User Interaction*, pages 169–169. ACM, 2016.

[70] W. Kienzle and K. P. Hinckley. Smart ring, Feb. 28 2017. US Patent 9,582,076.

[71] J. Kim, S. Yang, and M. Gerla. Stroketrack: wireless inertial motion tracking of human arms for stroke telerehabilitation. In *Proceedings of the First ACM Workshop on Mobile Systems, Applications, and Services for Healthcare*, page 4. ACM, 2011.

[72] G. Lan, W. Xu, S. Khalifa, M. Hassan, and W. Hu. Transportation mode detection using kinetic energy harvesting wearables. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–4. IEEE, 2016.

[73] A. Lazar, C. Koehler, J. Tanenbaum, and D. H. Nguyen. Why we use and abandon smart devices. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 635–646. ACM, 2015.

[74] A. Lazaro, R. Villarino, and D. Girbau. A survey of nfc sensors based on energy harvesting for iot applications. *Sensors*, 18(11):3746, 2018.

[75] H. V. Le, V. Schwind, P. Göttlich, and N. Henze. Predictouch: A system to reduce touchscreen latency using neural networks and inertial measurement units. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, pages 230–239. ACM, 2017.

[76] H.-K. Lee and J.-H. Kim. An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 21(10):961–973, 1999.

[77] X. Li, D. Zhang, Q. Lv, J. Xiong, S. Li, Y. Zhang, and H. Mei. Indotrack: Device-free indoor human tracking with commodity wi-fi. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3), Sept. 2017.

[78] Y. Li, T. Li, R. A. Patel, X.-D. Yang, and X. Zhou. Self-powered gesture recognition with ambient light. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, pages 595–608. ACM, 2018.

[79] K. Lin, J. Yu, J. Hsu, S. Zahedi, D. Lee, J. Friedman, A. Kansal, V. Raghunathan, and M. Srivastava. Heliomote: Enabling long-lived sensor networks through solar energy harvesting. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05. ACM, 2005.

[80] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[81] L. Liu, R. Zhang, and K.-C. Chua. Multi-antenna wireless powered communication with energy beamforming. *IEEE Transactions on Communications*, 62(12):4349–4361, 2014.

[82] X. Liu. Qi standard wireless power transfer technology development toward spatial freedom. *IEEE Circuits and Systems Magazine*, 15(2):32–39, Secondquarter 2015.

[83] Y. Liu, Z. Li, Z. Liu, and K. Wu. Real-time arm skeleton tracking and gesture inference tolerant to missing wearable sensors. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 287–299. ACM, 2019.

[84] H. J. Luinge, P. H. Veltink, and C. T. Baten. Estimation of orientation with gyroscopes and accelerometers. In *Proceedings of the First Joint BMES/EMBS Conference. 1999 IEEE Engineering in Medicine and Biology 21st Annual Conference and the 1999 Annual Fall Meeting of the Biomedical Engineering Society (Cat. N*, volume 2, pages 844–vol. IEEE, 1999.

[85] I. S. MacKenzie and C. Ware. Lag as a determinant of human performance in interactive systems. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 488–493. ACM, 1993.

[86] J. Marshall. Smartphone sensing for distributed swim stroke coaching and research. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 1413–1416. ACM, 2013.

[87] P. Miao, P. Mitcheson, A. Holmes, E. Yeatman, T. Green, and B. Stark. Mems inertial power generators for biomedical applications. *Microsystem Technologies*, 12(10-11):1079–1083, 2006.

[88] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4207–4215, 2016.

[89] S. Naderiparizi, A. N. Parks, Z. Kapetanovic, B. Ransford, and J. R. Smith. Wispcam: A battery-free rfid camera. In *2015 IEEE International Conference on RFID (RFID)*, pages 166–173. IEEE, 2015.

[90] P. Nguyen, U. Muncuk, A. Ashok, K. R. Chowdhury, M. Gruteser, and T. Vu. Battery-free identification token for touch sensing devices. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 109–122. ACM, 2016.

[91] H. Ohashi, M. Al-Nasser, S. Ahmed, T. Akiyama, T. Sato, P. Nguyen, K. Nakamura, and A. Dengel. Augmenting wearable sensor data with physical constraint for dnn-based human-action recognition. In *ICML 2017 Times Series Workshop, Sydney, Australia*, pages 6–11, 2017.

[92] S. J. Page, P. Levine, S. Sisto, Q. Bond, and M. V. Johnston. Stroke patients' and therapists' opinions of constraint-induced movement therapy. *Clinical rehabilitation*, 16(1):55–60, 2002.

[93] J. A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. *IEEE Pervasive computing*, (1):18–27, 2005.

[94] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 149–161. ACM, 2014.

[95] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 149–161. ACM, 2014.

[96] T. Park, J. Lee, I. Hwang, C. Yoo, L. Nachman, and J. Song. E-gesture: a collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 260–273. ACM, 2011.

[97] A. N. Parks, A. P. Sample, Y. Zhao, and J. R. Smith. A wireless sensing platform utilizing ambient rf energy. In *2013 IEEE Topical Conference on Biomedical Wireless Technologies, Networks, and Sensing Systems*, pages 154–156. IEEE, 2013.

[98] K. Pietroszek, L. Tahai, J. R. Wallace, and E. Lank. Watchcasting: Freehand 3d interaction with off-the-shelf smartwatch. In *3D User Interfaces (3DUI), 2017 IEEE Symposium on*, pages 172–175. IEEE, 2017.

[99] Pingskills. Pinkskills. `https://www.pingskills.com/tutorials/advanced-strokes/forehand-chop`, 2017. [Online; accessed 15-November-2017].

[100] Q. Pu, S. Gupta, S. Gollakota, and S. Patel. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 27–38. ACM, 2013.

[101] M. Radhakrishnan, A. Smailagic, B. French, D. P. Siewiorek, and R. K. Balan. Design and assessment of myoelectric games for prosthesis training of upper limb amputees. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 151–157. IEEE, 2019.

[102] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *Aaai*, volume 5, pages 1541–1546, 2005.

[103] D. Roetenberg, H. Luinge, and P. Slycke. Xsens mvn: Full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*, 1, 2009.

[104] D. Roetenberg, H. J. Luinge, C. T. Baten, and P. H. Veltink. Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation. *IEEE Transactions on neural systems and rehabilitation engineering*, 13(3):395–405, 2005.

[105] E. Rozner, V. Navda, R. Ramjee, and S. Rayanchu. Napman: Network-assisted power management for wifi devices. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10. ACM, 2010.

[106] K. Ryokai, P. Su, E. Kim, and B. Rollins. Energybugs: Energy harvesting wearables for children. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, 2014. ACM.

[107] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith. Design of an rfid-based battery-free programmable sensing platform. *IEEE Transactions on Instrumentation and Measurement*, 57(11):2608–2615, Nov 2008.

[108] A. Santos-Lozano, A. Hernández-Vicente, R. Pérez-Isaac, F. Santín-Medeiros, C. Cristi-Montero, J. A. Casajús, and N. Garatachea. Is the sensewear armband accurate enough to quantify and estimate energy expenditure in healthy adults? *Annals of translational medicine*, 5(5), 2017.

[109] T. Schlömer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 11–14. ACM, 2008.

[110] R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE transactions on antennas and propagation*, 34(3):276–280, 1986.

[111] S. Sen, A. Misra, V. Subbaraju, K. Grover, M. Radhakrishnan, R. K. Balan, and Y. Lee. I 4 s: capturing shopper's in-store interactions. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, pages 156–159. ACM, 2018.

[112] S. Sen, V. Subbaraju, A. Misra, R. K. Balan, and Y. Lee. The case for smartwatch-based diet monitoring. In *Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on*, pages 585–590. IEEE, 2015.

[113] S. Shen, H. Wang, and R. Roy Choudhury. I am a smartwatch and i can track my user's arm. In *Proceedings of the 14th annual international conference on Mobile systems, applications, and services*, pages 85–96. ACM, 2016.

[114] Y. K. Shetty. *Robust Human Motion Tracking Using Low-cost Inertial Sensors*. PhD thesis, Arizona State University, 2016.

[115] S. Siddhpuria, S. Malacria, M. Nancel, and E. Lank. Pointing at a distance with everyday smart devices. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 173. ACM, 2018.

[116] M. Sim and J.-U. Kim. Differences between experts and novices in kinematics and accuracy of golf putting. *Human movement science*, 29(6):932–946, 2010.

[117] StereoLabs. Zed stereo camera. `https://www.stereolabs.com/`, 2017. [Online; accessed 15-November-2017].

[118] T. Stiefmeier, D. Roggen, and G. Troster. Gestures are strings: efficient online gesture spotting and classification using string matching. In *Proceedings of the ICST 2nd international conference on Body area networks*, page 16. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[119] V. Talla, B. Kellogg, B. Ransford, S. Naderiparizi, S. Gollakota, and J. R. Smith. Powering the next billion devices with wi-fi. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, page 4. ACM, 2015.

[120] E. Taub, G. Uswatte, R. Pidikiti, et al. Constraint-induced movement therapy: a new family of techniques with broad application to physical rehabilitation-a clinical review. *Journal of rehabilitation research and development*, 36(3):237–251, 1999.

[121] E. Thomaz, I. Essa, and G. D. Abowd. A practical approach for recognizing eating moments with wrist-mounted inertial sensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1029–1040. ACM, 2015.

[122] E. Thomaz, I. Essa, and G. D. Abowd. A practical approach for recognizing eating moments with wrist-mounted inertial sensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 1029–1040. ACM, 2015.

[123] Z. Tian, J. Wang, X. Yang, and M. Zhou. Wicatch: a wi-fi based hand gesture recognition system. *IEEE Access*, 6:16911–16923, 2018.

[124] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. *arXiv preprint arXiv:1706.00527*, 2017.

[125] D. Vasisht, S. Kumar, and D. Katabi. Decimeter-level localization with a single wifi access point. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, pages 165–178. USENIX Association, 2016.

[126] E. Velloso, M. Carter, J. Newn, A. Esteves, C. Clarke, and H. Gellersen. Motion correlation: Selecting objects by matching their movement. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(3):22, 2017.

[127] T. H. Vu, A. Misra, Q. Roy, K. C. T. Wei, and Y. Lee. Smartwatch-based early gesture detection 8 trajectory tracking for interactive gesture-driven applications. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):39, 2018.

[128] R. Y. Wang and J. Popović. Real-time hand-tracking with a color glove. *ACM transactions on graphics (TOG)*, 28(3):63, 2009.

[129] Weka. Weka 3: Data Mining Software in Java. `https://www.cs.waikato.ac.nz/ml/weka/downloading.html`, 2017. [Online; accessed 15-November-2017].

[130] F. Wittmann, O. Lambercy, and R. Gassert. Magnetometer-based drift correction during rest in imu arm motion tracking. *Sensors*, 19(6):1312, 2019.

[131] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168. ACM, 2007.

[132] J. Xiong and K. Jamieson. Arraytrack: A fine-grained indoor location system. In *10th Usenix Symposium on Networked Systems Design and Implementation*. USENIX, 2013.

[133] G. Xu, Y. Yang, Y. Zhou, and J. Liu. Wearable thermal energy harvester powered by human foot. *Frontiers in Energy*, 7(1), 2013.

[134] Z. Yang, Z. Zhou, and Y. Liu. From rssi to csi: Indoor localization via channel response. *ACM Computing Surveys (CSUR)*, 46(2):25, 2013.

[135] D. J. Yeager, P. S. Powledge, R. Prasad, D. Wetherall, and J. R. Smith. Wirelessly-charged uhf tags for sensor data collection. In *2008 IEEE International Conference on RFID*, April 2008.

[136] E. M. Yeatman, P. D. Mitcheson, and A. S. Holmes. Micro-engineered devices for motion energy harvesting. In *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*, pages 375–378. IEEE, 2007.

[137] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[138] Z. Zhang, B. Halkon, S. M. Chou, and X. Qu. A novel phase-aligned analysis on motion patterns of table tennis strokes. *International Journal of Performance Analysis in Sport*, 16(1):305–316, 2016.

[139] J. Zhao and Z. You. A shoe-embedded piezoelectric energy harvester for wearable sensors. *Sensors*, 14(7):12497–12510, 2014.

[140] Y. Zheng, K.-C. Chan, and C. C. Wang. Pedalvatar: An imu-based real-time body motion capture system using foot rooted kinematic model. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4130–4135. IEEE, 2014.

[141] H. Zhou, T. Stone, H. Hu, and N. Harris. Use of multiple wearable inertial sensors in upper limb motion tracking. *Medical engineering & physics*, 30(1):123–133, 2008.

[142] ZoomTT. The Fastest Sport. `https://zoomtt.com/2017/04/15/fastest-sport-table-tennis-vs-badminton/`, 2017. [Online; accessed 15-May-2017].